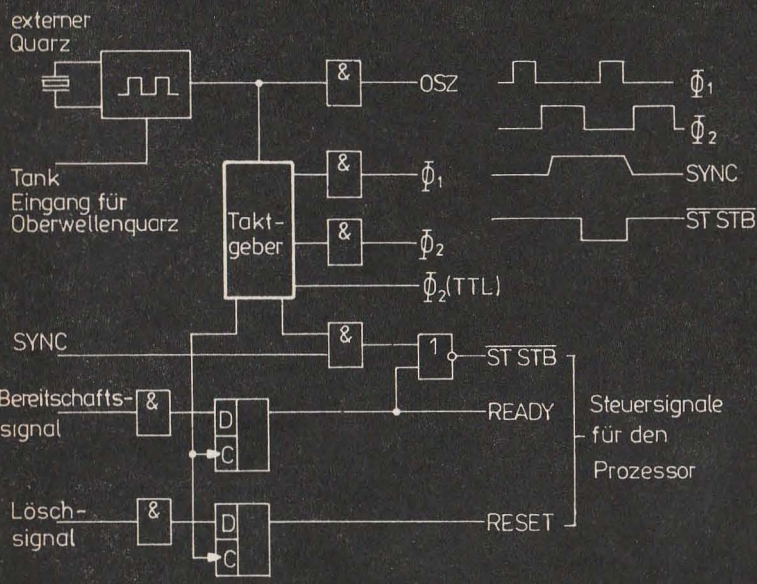
amateurreihe

electronica



Barthold/Bäurich

186

Mikroprozessoren — Mikroelektronische Schaltkreise und ihre Anwendung (Teil 1)

electronica · Band 186

HANS BARTHOLD
DR. HEINZ BÄURICH

Mikroprozessoren – Mikroelektronische Schaltkreise und ihre Anwendung

Teil 1: Grundlagen der Mikrorechentechnik

MILITÄRVERLAG
DER DEUTSCHEN DEMOKRATISCHEN
REPUBLIK

1. Auflage, 1980, 1.—20. Tausend
© Militärverlag der
Deutschen Demokratischen Republik (VEB) — Berlin, 1980
Cheflektorat Militärliteratur
Lizenz-Nr. 5 · LSV: 3539
Lektor: Dipl.-Phys. Hans-Joachim Mönig
Zeichnungen: Johanna Goernemann
Typografie: Helmut Herrmann · Hersteller: Michael Haase
Vorauskorrektor: Ilse Fähndrich · Korrektor: Gertraut Purfürst
Printed in the German Democratic Republic
Gesamtherstellung: Druckerei Märkische Volksstimme Potsdam
Redaktionsschluß: 21. April 1980
Bestellnummer: 746 216 5
DDR 1,90 M

Inhaltsverzeichnis

Vorwort		5
1.	**Erläuterungen zu den Abkürzungen**	7
1.1.	Aufstellung der verwendeten Formelzeichen und Abkürzungen	7
1.2.	Wertzuweisung bei logischen Signalen	7
2.	**Grundlagen der Rechentechnik**	9
2.1.	Aufbau eines Rechners	9
2.1.1.	Speicher	10
2.1.2.	Rechenwerk	11
2.1.3.	Steuerwerk	12
2.2.	Darstellung von Daten (Zahlen und Text)	14
2.2.1.	Zahlendarstellung	14
2.2.1.1.	Ganze Zahlen im Dualsystem	14
2.2.1.2.	Gebrochene Zahlen im Dualsystem	16
2.2.1.3.	Darstellung von negativen Zahlen	17
2.2.1.4.	Mathematische Formulierung der Komplementbildung	18
2.2.1.5.	Zahlensysteme mit der Basis 8 (Oktalsystem) und 16 (Hexadezimalsystem)	19
2.2.1.6.	Zifferncode	19
2.2.1.7.	Zahlenbereich im Rechner	22
2.2.2.	Darstellung von Text (Zeichencode)	23
2.3.	Aufbau der Befehle	24
2.4.	Befehlsschlüssel eines Rechners	25
2.5.	Befehlsabarbeitung	27
2.6.	Ein- und Ausgabesteuerung (E/A-Steuerung)	29
2.6.1.	Prinzip der E/A-Steuerung	29
2.6.2.	Programmierte Ein- und Ausgabe	31
2.6.3.	Autonome Ein- und Ausgabe	31
2.7.	Programmunterbrechung (INTERRUPT)	34
2.8.	STACK-Organisation	35
2.9.	Zusammenstellung der Funktionseinheiten eines Rechners	36

3. Aufbau und Arbeitsweise digitaler Schaltkreise für Mikrorechner 37
3.1. Übersicht 37
3.2. Schaltalgebra 38
3.3. Logische Grundschaltkreise 45
3.4. Informationsspeicherung 46
3.4.1. Überblick 46
3.4.2. Register 47
3.4.3. Speicher 54
3.4.4. Zusammenstellung einiger Speicherschaltkreise . 70
3.5. Codier- und Decodierschaltungen 76
3.6. Rechenschaltkreise 88
3.7. Bustreiber 96
3.8. Zähler 98
3.9. Taktgeneratoren 101

Anhang

1: Darstellung der Zahlen 0 bis 32 in dezimaler, dualer, oktaler, hexadezimaler und BCD-Form 105
2: Rechnen mit Dualzahlen 106
3: ASCII-Zeichencode 114
4: SIF-1000-Zeichencode 115

Literaturverzeichnis 116

Vorwort

Der Mikroprozessor, der vor 10 Jahren noch nicht einmal existierte, ist sehr schnell zu einem Begriff geworden, der aus der Elektronik nicht mehr wegzudenken ist. In der Fachliteratur spricht man in diesem Zusammenhang von Bausteinen der künftigen Automatisierungstechnik. Selbst solche Aussagen wie „Nervensystem der Technik" werden im Zusammenhang mit dem Mikroprozessor gebraucht. Unsere Massenmedien berichten in Wort und Bild, daß durch den Mikroprozessor ein Teil der Arbeitswelt völlig verändert wird. Während in den kapitalistischen Ländern die Rationalisierung mit Hilfe dieser Elektronik dazu führt, daß Arbeitsplätze abgeschafft werden und dadurch die Arbeitslosigkeit weiter steigt, dient der Einsatz der Mikrorechentechnik in den sozialistischen Ländern zum Nutzen der gesamten Gesellschaft und somit dem Wohl jedes einzelnen.

Aus allen diesen Informationen heraus drängen sich die Fragen auf, was ist ein Mikroprozessor, was kann er wirklich leisten? Das gesamte Gebiet der Elektronik hat sich seit 1945 sehr rasch entwickelt. Eine wesentliche Rolle dabei spielt die Entwicklung der Bauelemente. Von den anfänglichen Bauelementen der Informationstechnik bis zu den hochintegrierten Schaltkreisen haben sich Platz- und Energiebedarf für ein aktives Bauelement sehr stark verringert, gleichzeitig sind Zuverlässigkeit und Betriebssicherheit in sehr hohem Maße angestiegen.

Alle Gebiete der Elektronik wurden von dieser schnellen Bauelementeentwicklung beeinflußt. Dabei ist auf dem Gebiet der elektronischen Rechentechnik diese Entwicklung besonders sichtbar. Aus diesem Fachgebiet kommen auch die programmierbaren Schaltkreise, wobei der bekannteste Vertreter der Mikroprozessor ist. Er stellt den eigentlichen Rechenschaltkreis dar; mit ihm lassen sich alle Verknüpfungen von Zahlen und logischen Größen realisieren.

Die Verknüpfung solcher Größen bildet gleichzeitig die Grundlage der Automatisierungstechnik. Im Grunde genommen können mit jedem elektronischen Rechner die Probleme der Automatisierung gelöst werden. Aber erst durch die Möglichkeit, den Rechner als

Bauelement (Chip) herzustellen, wurde die stürmische Entwicklung in der Automatisierungstechnik erreicht. Es läßt sich einschätzen, daß diese Entwicklung noch am Anfang steht. Aber schon in den nächsten Jahren wird der Mikroprozessor in vielen technischen Geräten eine Selbstverständlichkeit sein.

Die 3 Bände dieses Titels sollen dem Leser das Wissen vermitteln, das er benötigt, um mit dem Mikroprozessor und anderen LSI-Schaltkreisen umgehen zu können. Da mit dem Mikroprozessor eine neue Denkweise in der Elektronik einhergeht, war es notwendig, zunächst im ersten Band die Grundlagen der Mikrorechentechnik darzulegen. Im 2. Teil stehen einige Mikroprozessorschaltkreise im Mittelpunkt. Im 3. Teil werden periphere Rechnerschaltkreise sowie einfache Programmbeispiele vorgestellt.

Die 3 Teile sind als eine Einheit zu betrachten und bauen aufeinander auf. Dabei findet der Anfänger alle Begriffe und Grundlagen, die er zum Verständnis eines Mikroprozessors benötigt.

Berlin, im Frühjahr 1980

Hans Barthold
Dr. Heinz Bäurich

1. Erläuterungen zu den Abkürzungen

1.1. Aufstellung der verwendeten Formelzeichen und Abkürzungen

Nachstehend sind die verwendeten Formelzeichen und Abkürzungen aufgeführt. Spezielle Zeichen, die nur wenig benutzt werden, sind im Text erläutert.

B	Zahlenbasis
E	Exponent einer Gleitkommazahl
H	hoher logischer Spannungspegel
L	niedriger logischer Spannungspegel
m	Mantisse einer Gleitkommazahl
P_j	Elementarkonjunktion; die der Dualzahl j zugeordnet ist
T	logischer Term (logischer Ausdruck)
Q, $\overline{Q}$	Ausgänge eines Flip-Flop
V	Vorzeichen einer Zahl
X_i	Ziffern einer Zahl
X (Y)	Wert einer Zahl
X'	Zweierkomplement der Zahl X

1.2. Wertzuweisung bei logischen Signalen

In den verwendeten Schaltbildern sind die logischen Signale durch Abkürzungen eingezeichnet. Die Abkürzungen werden im Text erläutert. Ist am Eingang oder am Ausgang eines Bausteins ein logisches Signal S durch S gekennzeichnet, heißt das: Das Signal hat Hochpegel, wenn dem Signal der logische Wert 1 zugeordnet ist. Wird es durch $\overline{S}$ gekennzeichnet, heißt das: Das Signal hat Tiefpegel, wenn dem Signal der logische Wert 1 zugeordnet ist. In den Funktionstabellen bedeuten die Bezeichnungen H bzw. L hoher bzw. niedriger Spannungspegel. Dagegen kennzeichnen 1 oder 0 die den Eingängen bzw. Ausgängen zugeordneten logischen Werte. Die Bezeichnung H und L wurde in den Fällen gewählt, wo es sich um die Beschreibung der Signalpegel in einer Schaltung handelt. 0 und 1 stehen in den Fällen, wo aus vorgege-

benen logischen Bedingungen das logische Schaltbild entworfen wird, ohne daß dabei die Signalzuordnung H oder L erforderlich ist.

2. Grundlagen der Rechentechnik

2.1. Aufbau eines Rechners

Der Rechner war das Vorbild für die Entwicklung der Mikroprozessoren. Viele Vorgänge in Natur und Technik laufen nach mathematischen oder logischen Regeln ab. Der Rechner ist die Basis zur Nachbildung solcher Vorgänge. Durch die Möglichkeit, ihn zu programmieren, kann man in ihm Algorithmen speichern und zu jeder Zeit abarbeiten lassen.
Der Rechner ist ein universelles Hilfsmittel zur Realisierung von Steuerungen, zur Nachbildung von Modellen sowie zur Lösung von mathematischen Aufgaben, für die man sich bisher umfangreicher elektronischer Schaltungen bedienen mußte (z. B. Digitaluhr, Digitalvoltmeter, Zähler). Durch seine Programmierbarkeit hat er gegenüber anderen Lösungen den Vorteil, daß er sich ohne Änderung der *Hardware* (fest verdrahtete Schaltung) an das jeweilige Problem anpassen läßt. Sein Einsatz für eine bestimmte Lösung hängt im Prinzip nur vom Kostenverhältnis zwischen Rechner und konkreter Schaltungstechnik ab.
Ein Rechner soll eine Aufgabe nach einer Lösungsvorschrift abarbeiten. Dabei ist der Ablauf derselbe wie bei einer Handrechnung. Entsprechend der Lösungsvorschrift, die aus einer Reihe von Anweisungen besteht, werden die Daten durch Rechenoperationen miteinander verknüpft, eventuelle Zwischenergebnisse notiert und die Ergebnisse auf einem gesonderten Formular zusammengefaßt.
In der gleichen Weise arbeitet ein Rechner. Die Lösungsvorschrift, die aus einer Reihe von Anweisungen (*Befehlen*) besteht, ist das *Programm*. Man muß es, damit es abgearbeitet werden kann, in einen Speicher eingeben. Ebenso müssen die Ausgangsdaten, die Zwischenwerte sowie die Resultate gespeichert werden. Das *Rechenwerk*, das im wesentlichen die 4 Grundrechenarten Addition, Subtraktion, Multiplikation und Division sowie die logischen Operationen UND, ODER und NEGATION ausführt, verknüpft die Daten. Für die Steuerung des gesamten Ablaufs gibt es ein *Steuerwerk*. Dieses Steuerwerk liest Anweisung für An-

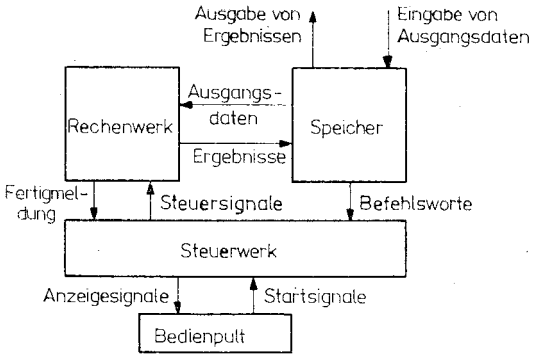


Bild 2.1 Grundsätzliche Struktur eines Rechners

weisung der Lösungsvorschrift aus dem Speicher und gibt dem Rechenwerk Signale zur Ausführung der in der Anweisung vorgegebenen Funktion. Rechenwerk und Steuerwerk bilden die CPU (*C*entral *P*rozessor *U*nit). Ein Rechner besteht also aus den Hauptbestandteilen Speicher, Rechenwerk und Steuerwerk.
Außerdem gehören zum Rechner noch Ein- und Ausgabegeräte zur Eingabe des Programms und der Ausgangsdaten einer Aufgabe sowie zur Ausgabe der Ergebnisse. Ferner ist an jedem Rechner ein Bedienpult angeschlossen, über das bestimmte Funktionen (Starten eines Programms, Ein- und Ausschalten von Teilgeräten, Sichtanzeigen) gesteuert werden können.
Bild 2.1 zeigt die grundsätzliche Struktur eines solchen Rechners.

2.1.1. Speicher

Der Speicher hat die Aufgabe, alle Informationen, die zur Lösung einer Aufgabe benötigt werden, zu speichern.
Dazu gehören das Programm, das aus einer Folge von Befehlen besteht, die Ausgangsdaten, die Zwischenresultate und die Endergebnisse.
Ausgangsdaten, Zwischenresultate und Endergebnisse sind im allgemeinen Zahlen oder alphanumerische Zeichen (Text). Zur Darstellung einer Zahl oder eines Befehls dient innerhalb des Rechners ein sogenanntes *Maschinenwort*, das aus einer Bit-Folge

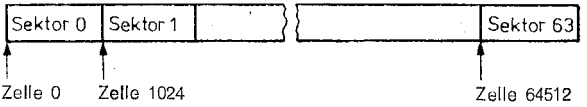


Bild 2.2 Logische Aufteilung eines 64-K-Speichers in 64 Sektoren

besteht. Es hat eine vorgegebene Wortlänge. Den Platz, der notwendig ist, um ein solches Wort im Rechner zu speichern, nennt man eine *Speicherzelle.* Der Speicher besteht aus einer größeren Anzahl solcher Speicherzellen, die durchnumeriert sind. Die Nummer der Speicherzelle nennt man die *Adresse.*

In der Rechentechnik ist die Mengenangabe Kilo (Kilobyte oder Kiloworte) üblich. Ein Kilo kennzeichnet hier jedoch nicht die tausendfache Menge, sondern das 1024fache (1 Kbyte = 1024 Byte), da die Kapazität eines Speichers fast immer einer Zweierpotenz entspricht. Zu einer solchen Speicherplatzanzahl ist eine optimale Adreßentschlüsselung möglich. Manchmal wird der Speicher in Seiten oder Sektoren aufgeteilt. Entsprechend unterteilt sich dann auch die Adresse in Seitenadresse (Sektoradresse) und Zellennummer innerhalb der Seite (Sektor). Bild 2.2 zeigt die Aufteilung eines Speichers mit 64 K-Zellen in 64 Sektoren.

Technisch werden Speicher meistens als Ferritkernspeicher oder Halbleiterspeicher ausgeführt (s. Abschnitt 3.4.3.).

2.1.2. Rechenwerk

Das Rechenwerk dient zur Ausführung von Rechenoperationen. Es bekommt durch das Steuerwerk eine Folge von Schaltsignalen, die es auf die gerade auszuführenden Rechenoperationen umschalten. Vom Speicher erhält es die Zahlen, die zur Ausführung der Rechenoperationen notwendig sind.

Die Zahlen werden im Rechenwerk zwischengespeichert. Dazu hat das Rechenwerk mehrere Register, in denen die Ausgangsdaten einer Rechenoperation gespeichert sind. Nach der Ausführung der Rechenoperation wird das Ergebnis ebenfalls in einem Register gespeichert. Bild 2.3 zeigt das Grundprinzip eines Rechenwerks.

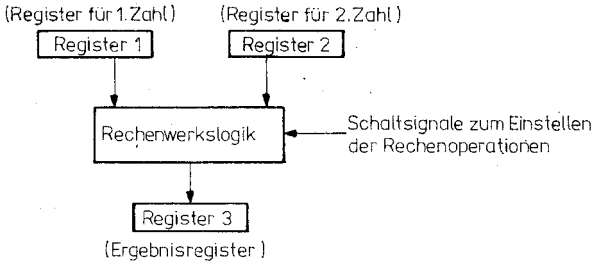


Bild 2.3 Grundprinzip eines Rechenwerks

Zu den Operationen, die ein Rechenwerk eines Rechners ausführt, gehören:

a) Arithmetische Rechenoperationen für die im Rechner verdrahteten Zahlendarstellungen (ADDITION, SUBTRAKTION, MULTIPLIKATION und DIVISION). Bei Mikrorechnern sind im allgemeinen nur Addition und Subtraktion möglich. Multiplikation und Division müssen programmiert werden.
b) Logische Operationen (UND, ODER, EXKLUSIV-ODER, NEGATION).
c) VERSCHIEBUNG.
d) Zahlenumwandlungen.

Das Rechenwerk eines Mikrorechners wird im allgemeinen als *arithmetisch-logische Einheit* (ALU — *a*rithmetic-*l*ogic *u*nit) bezeichnet. Die ALU ist eine Logikschaltung, die durch Steuersignale so eingestellt werden kann, daß ein oder zwei Eingangsbitmuster entsprechend der eingestellten Operation verarbeitet werden.

2.1.3. Steuerwerk

Das Steuerwerk übernimmt die Befehle eines Programms in der vorgegebenen Reihenfolge aus dem Speicher, entschlüsselt sie und bildet daraus die Steuersignale für das Rechenwerk. Bild 2.4 zeigt den logischen Aufbau des Steuerwerks.

Der Befehlszähler enthält die Adresse des zu holenden Befehls. Der vom Speicher übernommene Befehl wird zur Befehlsentschlüsselung im Befehlsregister zwischengespeichert. Das Spei-

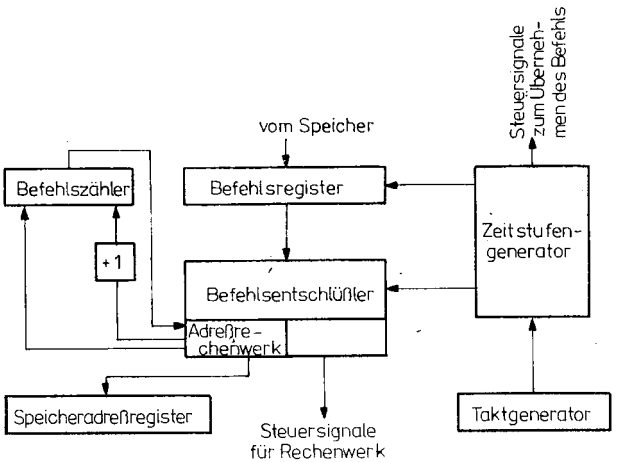


Bild 2.4 Aufbau des Steuerwerks eines Rechners

cheradreßregister enthält die Adresse des zum Befehl benötigten Operanden. (Mehrere Operanden [Zahlen] werden nacheinander aus dem Speicher geholt.)

Der Zeitstufengenerator erzeugt eine Reihe von Zeitsignalen, die den Ablauf der Befehlsabarbeitung festlegen. Den Grundtakt dazu liefert der Taktgenerator. Die wichtigsten Schritte einer Befehlsabarbeitung sind:

— Übernehmen des Befehls aus dem Speicher;
— Entschlüsseln des Befehls;
— Ermitteln der Operandenadresse;
— Holen der Operanden aus dem Speicher;
— Ausführung des Befehls;
— Ermittlung der Adresse des nächsten Befehls.

Da der Rechner aus Speicher, Rechen- und Steuerwerk besteht, ergibt sich die Möglichkeit, automatisch, d. h. programmgesteuert, beliebige Funktionen oder Aufgaben abzuarbeiten. Zu diesem Zweck enthält jede Aufgabe 2 Teilinformationen. Die erste Teilinformation beinhaltet die Daten bzw. Zahlen, die verarbeitet werden sollen. Der zweite Teil der Information sind die Anweisungen (Befehle), die aussagen, wie die Daten verarbeitet werden sollen. Beide Teilinformationen gibt man über externe Geräte (Eingabetastatur, Schreibmaschine, Lochbandleser, Loch-

karteneingabe usw.) in den Speicher ein. Vom Speicher werden die Anweisungen in einer festgelegten Reihenfolge ins Steuerwerk übernommen, entschlüsselt und vom Rechenwerk abgearbeitet. Zwischen- und Endresultate gelangen in den Speicher zurück und können von dort aus wieder über externe Geräte (Anzeige, Schreibmaschine, Drucker) ausgegeben werden.

2.2. Darstellung von Daten (Zahlen und Text)

Da der Speicher aus einzelnen Zellen besteht, wobei jede Zelle eine Bit-Folge mit vorgegebener Länge speichern kann, müssen die zu verarbeitenden Daten (Zahlen) in Bit-Folgen umgewandelt werden. Dafür gibt es eine Reihe von Darstellungsarten. Je nach Länge einer Speicherplatzzelle und Anzahl der notwendigen Stellen für die Darstellung einer Zahl werden für die Speicherung einer Zahl eine oder mehrere Zellen benötigt. Das gleiche gilt für die Speicherung von alphanumerischem Text.

2.2.1. Zahlendarstellung

Ausgangspunkt für die Darstellung von Zahlen in Rechnern ist das *Dualsystem*. Dabei handelt es sich um ein Zahlensystem, das auf nur 2 Ziffern basiert (0 und 1). Zur Speicherung einer Dualziffer sind also nur 2 stabile Zustände notwendig, die sich leicht realisieren lassen.

2.2.1.1. Ganze Zahlen im Dualsystem

Um eine Dezimalzahl in das Dualsystem zu übertragen, zerlegt man sie in Potenzen zur Basis 2, wobei jeder neuen Potenz von 2 eine neue Stelle zugeordnet wird.

Beispiel
Die Zahl 27 soll dual dargestellt werden. Aus Tabelle 2.1. ist zu ersehen, daß die größte in 27 enthaltene Zweierpotenz $2^4 = 16$ ist, d. h., der Stellenwert 2^4 wird benötigt. Der Stellenwert $2^3 = 8$ ergibt mit 2^4 zusammen 24, der Rest ist 3. Der Stellenwert 2^2 tritt also nicht, d. h. 0mal, auf, während die Stellenwerte 2^1 und

Tabelle 2.1. Tabelle der Zweierpotenzen

2^n	n	2^{-n}
1	0	1
2	1	0,5
4	2	0,25
8	3	0,125
16	4	0,0625
32	5	0,03125
64	6	0,015625
128	7	0,0078125
256	8	0,00390625

2^0 je einmal erforderlich sind. Die Zahl 27 stellt sich im Dualsystem also folgendermaßen dar:

Stelle 2^4	Stelle 2^3	Stelle 2^2	Stelle 2^1	Stelle 2^0
1	1	0	1	1

Der Leser führe selbst die Umwandlung der folgenden Dezimalzahlen ins Dualsystem aus:
$41 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \triangleq 101001$ (dual);
$15 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \triangleq 1111$ (dual).

Umwandlung Dezimal → Dual
Eine weitere Methode besteht in der fortlaufenden Division durch 2.

Beispiel
Es soll die Dezimalzahl 93 durch fortlaufende Division durch 2 dual dargestellt werden.

	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$93:2 = 46$ Rest 1							1
$46:2 = 23$ Rest 0						0	
$23:2 = 11$ Rest 1					1		
$11:2 = 5$ Rest 1				1			
$5:2 = 2$ Rest 1			1				
$2:2 = 1$ Rest 0		0					
$1:2 = 0$ Rest 1	1						
	1	0	1	1	1	0	1

93 (dezimal) $\triangleq$ 1011101 (dual).

Umwandlung Dual → Dezimal
Hier geht man von einem Rechenschema aus, in dem die 2er Potenzen wieder dezimal dargestellt werden.

Beispiel
$10111 \triangleq 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 0 + 4 + 2 + 1 = 23.$

2.2.1.2. Gebrochene Zahlen im Dualsystem

Bei der Darstellung von Zahlen, die Stellen hinter dem Komma haben, wird ähnlich wie bei den ganzen Zahlen verfahren. Man benötigt dazu eine Tabelle der Potenzen von 2 mit negativen Exponenten. Zum Beispiel kann man die Zahl 0,625 folgendermaßen schreiben:

$0{,}625 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \triangleq 0{,}101$ (dual).

Ein Dezimalbruch läßt sich ins Dualsystem durch fortlaufende Multiplikation mit der Grundzahl 2 überführen. Steht nach der Multiplikation mit 2 vor dem Komma eine 0, so ist die nächste Dualstelle eine 0. Steht eine 1 vor dem Komma, dann ist die nächste Dualstelle eine 1.

Beispiel
Umwandlung der Dezimalzahl 0,625
$0{,}625 \cdot 2 = 1{,}250 \quad 1 \triangleq$ 1. Dualstelle nach dem Komma;
$0{,}25 \cdot 2 = 0{,}50 \quad 0 \triangleq$ 2. Dualstelle nach dem Komma;
$0{,}50 \cdot 2 = 1{,}00 \quad 1 \triangleq$ 3. Dualstelle nach dem Komma.
Die Dezimalzahl 0,625 lautet dual 0,101. Die Begründung des Verfahrens liegt darin, daß bei jeder Multiplikation mit 2 die Zweierpotenzen um 1 erhöht werden und dabei die erste Ziffer vor das Komma rückt. Diese Ziffer wird als Dualstelle übernommen.
Bei gemischten Zahlen wandelt man den ganzen Teil und den gebrochenen Teil — jeden für sich — in eine Dualzahl um.

Beispiel
Die Zahl 3,25 soll dual dargestellt werden. Die Zerlegung von 3,25 ergibt 3 + 0,25. Der Zahl 3 entspricht die Dualzahl 11, der Zahl 0,25 entspricht die Dualzahl 0,01.
Damit lautet die Zahl 3,25 dual 11,01.

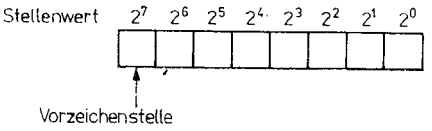


Bild 2.5
Darstellung von positiven und negativen ganzen Zahlen mit Hilfe von 8 Dualstellen

2.2.1.3. Darstellung von negativen Zahlen

In der Umgangssprache unterscheiden sich die negativen Zahlen von den positiven Zahlen durch ein negatives Vorzeichen. Für die Darstellung dieses Vorzeichens im Rechner kann man eine zusätzliche Dualstelle als Vorzeichenstelle einführen und z. B. folgende Vereinbarung treffen:
Vorzeichenstelle = 0 heißt positives Vorzeichen.
Vorzeichenstelle = 1 heißt negatives Vorzeichen.
Diese Darstellungsart wird in der Literatur mit „Betrag und Vorzeichen" bezeichnet.
In den meisten Rechnern wird jedoch eine andere Darstellung, die *Komplementdarstellung*, verwendet, in der die negativen Zahlen in den positiven Zahlenbereich transformiert werden. Der Vorteil besteht darin, daß sich die Subtraktion auf eine Addition zurückführen läßt.
In den meisten Mikroprozessoren stehen 8 Dualstellen (7 Dualstellen für die positiven Zahlen und eine Dualstelle für die Vorzeichenstelle) zur Verfügung (Bild 2.5). Werden diese 8 Dualstellen zur Darstellung ganzer Zahlen verwendet, so hat die Zahl +5 die Belegung in Zeile 1 von Bild 2.6. Die duale Darstellung der Zahl −5 mit Betrag und Vorzeichen ist aus Zeile 2 zu ersehen. Bei der Komplementdarstellung wird bei negativen Zahlen das Komplement zu 2^8 gebildet. Das heißt, im Rechner wird anstelle

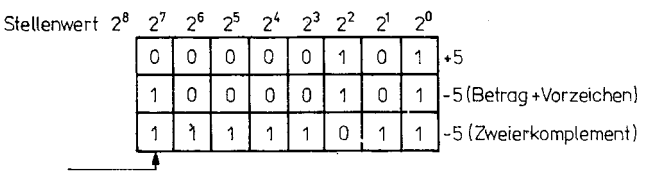


Bild 2.6 Darstellung der Zahlen + 5 und — 5 im Zweierkomplement

der Zahl $-x$ die Zahl $2^8 - x$ gespeichert. Für die Zahl -5 entsteht dabei die Belegung entsprechend Zeile 3 in Bild 2.6.
Für die Komplementbildung kann man folgende Regel aufstellen:
Ausgehend von der positiven Dualzahl, werden zur Komplementbildung alle Stellen negiert und anschließend zum niedrigsten Stellenwert eine 1 addiert.

2.2.1.4. Mathematische Formulierung der Komplementbildung

Der positive Zahlenbereich der nach Bild 2.7 dargestellten Zahlen ist

$$0 \leqq x \leqq \sum_{i=-M}^{N} 2^i = 2^{N+1} - 2^{-M}.$$

Da die negativen Zahlen in den positiven Zahlenbereich transformiert werden sollen, gilt für die Zahlen x' im Rechner:

$$x' = \begin{cases} {}^+x, & \text{falls } x \geqq 0 \\ {}^+x + k, & \text{falls } x < 0 \end{cases},$$

wobei k eine Komplementärzahl ist. Für die Komplementärzahl k ergeben sich folgende Bedingungen:

1. Es soll $x' > 0$ sein.

2. Positive und negative Zahlen müssen unterscheidbar sein.

Damit muß für negative Zahlen x

$k + x > 2^{N+1} - 2^{-M}$

sein, d. h.

$k > 2^{N+1} - 2^{-M} - x = 2^{N+1} - 2^{-M} + |x|,$

und mit

$|x|_{\max}| = 2^{N+1} - 2^{-M}$

wird

$$k > 2 \cdot 2^{N+1} - 2 \cdot 2^{-M} = k'.$$

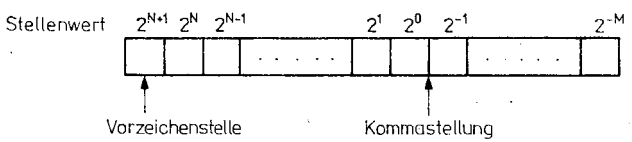


Bild 2.7 Duale Zahlendarstellung im Rechner

Die kleinstmögliche Komplementärzahl ist damit

$$k_1 = k' + 2^{-M} = 2^{N+2} - 2^{-M} .$$

Negative Zahlen, die mit dieser Komplementärzahl gebildet werden, heißen *Einerkomplement.*
Die nächstmögliche Komplementärzahl ist:

$$k_2 = k_1 + 2^{-M} = 2^{N+2} .$$

Zahlen, die mit k_2 gebildet werden, nennt man *Zweierkomplement.* In unserem Beispiel nach Bild 2.5 sind $M = 0$ und $N = 6$ und damit $k_2 = 2^8$.

2.2.1.5. Zahlensysteme mit der Basis 8 (Oktalsystem) und 16 (Hexadezimalsystem)

Sehr häufig werden zur übersichtlichen Darstellung von Dualzahlen dem Dualsystem verwandte Zahlensysteme verwendet. Hierzu gehören das Oktalsystem und das Hexadezimalsystem.

Oktaldarstellung
Beim Oktalsystem beträgt die Basis $B = 8$. Da $8 = 2^3$ ist, bilden immer 3 Dualziffern eine Oktalziffer. Im Oktalsystem werden 7 Ziffern (0 bis 7) benötigt.

Beispiel
Die Dezimalzahl 201 lautet als Dualzahl 11001001 und als Oktalzahl 311 (11/001/001).

Hexadezimaldarstellung
Das Hexadezimalsystem hat die Basis $B = 16$. Da $16 = 2^4$ ist, bilden 4 Dualziffern eine Hexadezimalziffer. Im Hexadezimalsystem werden 16 Ziffern benötigt. Für die Ziffern (10), (11), (12), (13), (14), (15) setzt man gewöhnlich die Zeichen A, B, C, D, E, F ein.

Beispiel
Die Dezimalzahl 201 lautet als Hexadezimalzahl
C 9 (1100/1001).

2.2.1.6. Zifferncode

Um Zahlen binär auszudrücken, gibt es außer der reinen Dualdarstellung noch gemischte Formen, sogenannte Codierungen,

in denen die Ziffern einer Dezimalzahl getrennt durch Binärziffern dargestellt werden.
Der einfachste Code ist der *BCD-Code* (Binär-Code), in dem jede Ziffer durch die entsprechende Dualzahl dargestellt wird. Für eine Ziffer werden dabei 4 Dualstellen benötigt.

Beispiel

3791 $\triangleq$	0011	0111	1001	0001
	3	7	9	1

Die für eine Dezimalziffer notwendigen Dualstellen nennt man eine *Tetrade*. Mit diesen 4 Stellen lassen sich außer den Ziffern 0 bis 9 noch die Zahlen 10 bis 15 realisieren. Tritt innerhalb der Tetrade eine Kombination von Binärstellen auf, die einer der Zahlen 10 bis 15 entspricht (z. B. 1101 $\triangleq$ 13), so nennt man diese Kombination eine *Pseudotetrade*. **Pseudotetraden sind Kombinationen von Binärziffern, deren Wert keiner der Ziffern 0 bis 9 zugeordnet ist.**
Außer dem BCD-Code gibt es noch weitere Codierungsvorschriften für Ziffern, in denen andere Kombinationen von Binärziffern den Dualziffern 0 bis 9 zugeordnet sind.

3-Exzeß-Code
Codewert = Dualwert + 3.
Die 3-Exzeß-Code-Verschlüsselung ergibt für die Dezimalziffern 0 bis 9 folgende Zuordnung:

0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Aiken-Code
Codewert = Dualwert, falls Zahlenwert < 5,
Codewert = Dualwert + 6, falls Zahlenwert $\geqq 5$.

Der Aiken-Code ergibt für die Dezimalziffern 0 bis 9 folgende Zuordnung:

0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Außer der Darstellung von Dezimalziffern durch 4 Dualstellen gibt es noch solche mit mehr als 4 Dualstellen.
Beispiele solcher Codierungen sind der *Walking-Code* und der *Positionscode.*

Walking-Code (2-aus-5-Code)

0 = 0 0 0 1 1
1 = 0 0 1 0 1
2 = 0 0 1 1 0
3 = 0 1 0 1 0
4 = 0 1 1 0 0
5 = 1 0 1 0 0
6 = 1 1 0 0 0
7 = 0 1 0 0 1
8 = 1 0 0 0 1
9 = 1 0 0 1 0

Positionscode (1-aus-10-Code)

0 = 0 0 0 0 0 0 0 0 0 1
1 = 0 0 0 0 0 0 0 0 1 0
2 = 0 0 0 0 0 0 0 1 0 0
3 = 0 0 0 0 0 0 1 0 0 0
4 = 0 0 0 0 0 1 0 0 0 0
5 = 0 0 0 0 1 0 0 0 0 0
6 = 0 0 0 1 0 0 0 0 0 0
7 = 0 0 1 0 0 0 0 0 0 0
8 = 0 1 0 0 0 0 0 0 0 0
9 = 1 0 0 0 0 0 0 0 0 0

2.2.1.7. Zahlenbereich im Rechner

Bei der Darstellung von Zahlen im Rechner muß man die Tatsache berücksichtigen, daß ein Rechner nur eine bestimmte Stellenzahl hat. Stehen z. B. nur 7 Stellen für positive Dualzahlen zur Verfügung, wie es bei vielen Mikroprozessoren der Fall ist, so kann man damit nur ganze Zahlen zwischen 0 und 127 darstellen. Sollen auch gebrochene Zahlen dargestellt werden, so kann man das Komma vor einer dieser Stellen setzen. Dabei wird der Zahlenbereich nicht erweitert, sondern nur verschoben. Steht bei 7 Stellen das Komma nach der 4. Stelle, so kann man nur Zahlen zwischen 0 und 15,875 in Schritten zu 0,125 darstellen.
Damit der Zahlenbereich an die praktischen Erfordernisse angepaßt wird, gibt es noch verschiedene Darstellungsformen innerhalb eines Zahlensystems, die wichtigsten sind die *Festkomma-* und die *Gleitkommazahlen.*

Festkommazahlen
Dabei handelt es sich um Zahlen, bei denen eine feste Anzahl von Ziffern vor und nach dem Komma vereinbart wird. Die Stellung des Kommas und die Gesamtzahl der Ziffern hängt vom Rechnertyp ab. Es gibt auch Rechner, bei denen das Komma durch Tasten oder selbständig gesetzt wird.

Beispiel
In jedem Taschenrechner sind Dezimalzahlen der Form 3 7 2 8 3 · 6 1 3 üblich. In diesem Fall rechnet der Taschenrechner mit 8 Dezimalstellen in Festkommadarstellung.
Im Dualsystem besteht das gleiche Problem. In einem Rechner läßt sich wegen der technischen Gegebenheiten nur eine feste Anzahl von Dualstellen speichern. Dabei kann das Komma rechts von der niedrigsten Stelle (dann handelt es sich um ganze Zahlen) oder links vor der höchsten Stelle stehen (dann handelt es sich um echt gebrochene Zahlen), oder das Komma trennt einen ganzen und einen gebrochenen Teil.
Bei der Addition und Subtraktion von Festkommazahlen müssen die Zahlen so verschoben werden, daß die Kommas untereinander stehen. Bei Multiplikation und Division muß man die Stelle des Kommas besonders bestimmen.

Gleitkommazahlen
Jede Zahl Z läßt sich in folgender Form darstellen:

$$Z = m \cdot B^E$$

(z. B. $0{,}19 \cdot 10^{-18}$; $m = 0.19$; $E = -18$; $B = 10$).

Dabei nennt man m die *Mantisse*, E den *Exponenten*, und B ist die *Basis* des Zahlensystems.

Bei der dualen Darstellung von Gleitkommazahlen gilt für m bei den meisten Rechnern folgende Vorschrift:

1. $-1 < m < +1$.
2. Die 1. Stelle nach dem Komma soll nicht 0 sein.

Zahlen, die diesen Bedingungen genügen, nennt man *normalisierte Zahlen*. Durch diese Vorschrift werden die Mantisse m und der Exponent E eindeutig bestimmt. Es gibt aber auch Vorschriften für die Bildung der Mantisse m, die von der genannten Vorschrift abweichen.

Beispiel 1

Man schreibe die Zahl 25,211 als Dezimalzahl in Gleitkommadarstellung.

Lösung: $25{,}211 = 0{,}25211 \cdot 10^2$.

Die Mantisse lautet also $m = 0{,}25211$ und der Exponent $E = 2$.

Beispiel 2

Man schreibe die Zahl 4,25 als Gleitkommazahl in dualer Darstellung.

Lösung: $4{,}25 = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$.

Das ergibt die Dualzahl 100,01.

Als Gleitkommazahl entsprechend obiger Vorschrift wird daraus $0{,}10001 \cdot 2^3 = 0{,}10001 \cdot 2^{11}$.

Die Mantisse lautet also $m = 0{,}10001$ und der Exponent $E = 11$.

2.2.2. Darstellung von Text (Zeichencode)

Damit der Rechner beliebige Textinformationen speichern kann, müssen alphanumerische Zeichen auch durch Bit-Folgen (Folgen aus den Ziffern 0 und 1) dargestellt werden. Dabei ergibt sich die Frage, wieviel Dualstellen insgesamt zur Codierung aller vorkommenden Zeichen notwendig sind. Aus der Mathematik ist bekannt, daß mit n Dualstellen 2^n unterschiedliche Bit-Folgen,

bestehend aus den Ziffern 0 und 1, dargestellt werden können. Nimmt man an, daß die Anzahl der verwendeten Zeichen (Buchstaben, Ziffern und Syntaxzeichen) kleiner als 128 ist, so benötigt man dazu 7 Dualstellen, da $2^7 = 128$ ist. Meistens kommt zu diesen 7 Bit ein Kontrollbit hinzu. Die Bit-Folge von 8 Dualstellen nennt man 1 *Byte*. Innerhalb dieser 8 Stellen wird jedem Zeichen eine feste Folge aus den Ziffern 0 und 1 zugeordnet. Diese Folge ist der Code des betreffenden Zeichens.
Es gibt mehrere internationale Festlegungen für solche Zuordnungen. Die in der DDR am häufigsten verwendeten Zeichencode sind:

- *SIF-1000-Code* (Standard-Interface-Code für Datenverarbeitungsperipherie);
- ASCII-Code (American Standard Code for Information Interchange — amerikanischer Code für Informationsaustausch);
- R-300-Code;
- ISO-7-Bit-Code.

In Mikrorechnern wird vorwiegend der ASCII-Code und der SIF-1000-Code angewendet (s. Anhang 3).

2.3. Aufbau der Befehle

Genauso wie die Zahlen durch Bit-Folgen dargestellt werden, muß man auch die Anweisungen (Befehle) in Bit-Folgen umwandeln, um sie zu speichern. Da eine Anweisung (Befehl) eine Information sein soll, aus der hervorgeht, was der Rechner zu tun hat, besteht sie aus 2 Hauptteilen:

- Der 1. Teil sagt aus, was bzw. welche Operation der Rechner ausführen soll.
- Der 2. Teil sagt aus, woher die Daten kommen, die für die Durchführung dieser Operation gebraucht werden.

Der 1. Teil wird *Operationsteil*, der 2. Teil *Adreßteil* genannt (Bild 2.8). In Adreßteil steht eine Adresse (Einadreßbefehl) oder mehrere Adressen (Mehradreßbefehl) für die benötigten Daten. Im Adreßteil eines Einadreßbefehls steht entweder die fertige Adresse oder, wie es oft der Fall ist, eine Rechenvorschrift, aus der die endgültige Adresse ermittelt wird. Man spricht in diesem Fall

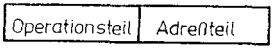


Bild 2.8 Befehlsaufbau

von *Adreßrechnung*. Durch die Adreßrechnung wird aus den im Adreßteil des Befehls stehenden Angaben die endgültige Speicheradresse ermittelt.

Bei der Adreßrechnung gibt es 2 Grundprinzipien, auf die bei nahezu allen Rechnern zurückgegriffen wird. Das erste Prinzip ist unter dem Namen *Index-Rechnung* bekannt. Bei der Index-Rechnung wird die endgültige Speicheradresse ermittelt, indem zu der im Befehl angegebenen Adresse der Inhalt eines speziellen Registers, *Index-Register* genannt, dazugezählt wird. Die endgültige Speicheradresse ist also gleich im Befehl angegebene Adresse plus Inhalt Index-Register (Indexierung):

Endgültige Speicheradresse = im Befehl angegebene Adresse + ⟨Index-Register⟩.

Das zweite Prinzip ist unter dem Namen *indirekte Adressierung* bekannt. Bei der indirekten Adressierung steht die endgültige Speicheradresse in der Zelle, deren Adresse im Befehl steht; oder anders ausgedrückt, die endgültige Adresse ist der Inhalt der im Befehl angegebenen Speicherzelle. Es kommt auch vor, daß die endgültige Adresse nicht in der Speicherzelle, sondern in einem speziellen Register steht.

Endgültige Speicheradresse = Inhalt der im Befehl angegebenen Speicherzelle.

2.4. Befehlsschlüssel eines Rechners

Der Befehlsschlüssel ist eine Zusammenfassung aller Befehle, die ein Rechner ausführen kann. Jedem Befehl wird eine bestimmte Bit-Folge im Operationsteil zugeordnet. Diese Bit-Folge nennt man *Operationscode*. Bei Befehlen, die keinen Speicherzugriff benötigen (*nichtspeicherbezogene Befehle*), wird zur Codierung des Befehls der Adreßteil hinzugezogen. Die folgende Zusammenstellung enthält eine Übersicht über Befehlsarten, die ein Mikroprozessor ausführen kann.

– *Adreßoperationen*

Adreßoperationen sind Operationen zur Ermittlung der endgül-

tigen Speicheradresse aus den Angaben im Adreßteil des Befehls (Indexierung, indirekte Adressierung).

– *Transportoperationen*

Transportoperationen dienen zur Übertragung von Daten vom Speicher in spezielle Register und umgekehrt oder von Speicherzellen in andere Speicherzellen. Man unterscheidet *Einzelworttransfer* und *Blocktransfer*. Einzelworttransfer ist die Übertragung eines Wortes oder Bytes, Blocktransfer ist die Übertragung eines Datenblocks von einem Speicherbereich in einen anderen Speicherbereich.

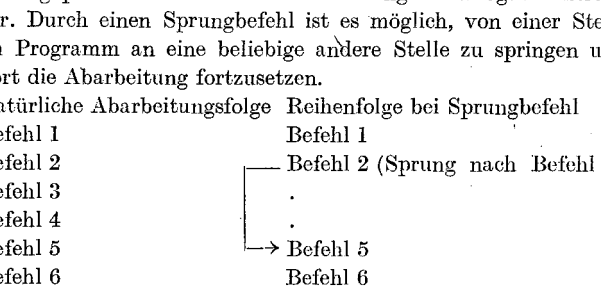
– *Rechen- und logische Operationen*

Diese Operationen dienen zur Verknüpfung von Zahlen bzw. Bit-Folgen. Bei den Rechenoperationen werden Zahlen durch Rechenvorschriften (Addition, Subtraktion, Multiplikation, Division), bei den logischen Operationen durch logische Operatoren (UND, ODER, NEGATION, EXKLUSIV-ODER usw.) verknüpft. Das Resultat steht meistens in einem speziellen Register, dem *Akkumulatorregister*.

Außerdem gibt es Befehle, durch die einzelne Bits einer Bit-Folge verändert werden, sowie Befehle zum Vergleich zweier Bit-Folgen.

Sprungoperationen

Sprungoperationen dienen zur Gestaltung der Programmstruktur. Durch einen Sprungbefehl ist es möglich, von einer Stelle im Programm an eine beliebige andere Stelle zu springen und dort die Abarbeitung fortzusetzen.



Unterprogrammbefehle

Ein Unterprogramm ist ein Programm, das eine spezielle Funktion ausführt, die während der Abarbeitung eines größeren Programms mehrmals notwendig ist (z. B. Berechnung des Logarithmus einer Zahl). Man spricht deshalb vom *Hauptprogramm*, das die gesamte Aufgabe realisiert, und vom *Unterprogramm* zur Lösung einer speziellen Funktion.

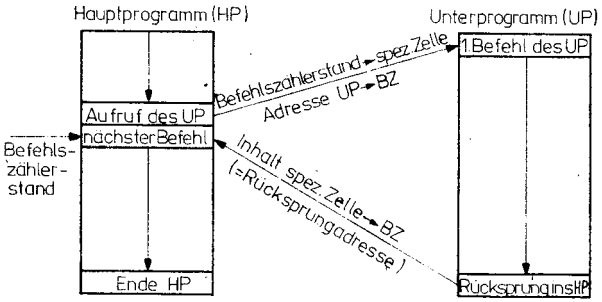


Bild 2.9 Arbeitsweise eines Unterprogramms

Das Unterprogramm steht nur einmal im Speicher und muß so arbeiten, daß man es von verschiedenen Stellen des Hauptprogramms aus aufrufen kann. Nach dem Durchlaufen des Unterprogramms muß an die Stelle im Hauptprogramm zurückgesprungen werden, von der aus ins Unterprogramm gesprungen wurde. Der Sprungbefehl ins Unterprogramm besteht aus 2 Funktionen. Erstens wird ein einfacher Sprung zur Startadresse des Unterprogramms ausgeführt, zweitens muß die Rücksprungstelle ins Hauptprogramm gemerkt werden. Die Rücksprungadresse (Inhalt Befehlszähler + 1) wird in einer speziellen Speicherzelle gemerkt.

Der Rücksprung aus dem Unterprogramm ist ein Sprungbefehl, dessen Adresse aus der Speicherzelle genommen wird, in der die Rücksprungadresse steht (Bild 2.9).

Steueroperation

Diese Operationen sind spezielle Befehle, die den Zustand des Rechners festlegen. Dazu gehört z. B. der HALT-Befehl.

Ein- und Ausgabeoperation

Sie dienen zur Ansteuerung entsprechender Kanäle, an die Geräte zur Ein- bzw. Ausgabe von Daten angeschlossen sind.

2.5. Befehlsabarbeitung

Zur Abarbeitung eines Programms werden die Befehle der Reihe nach aus dem Speicher in die CPU geholt. Die Adresse der Spei-

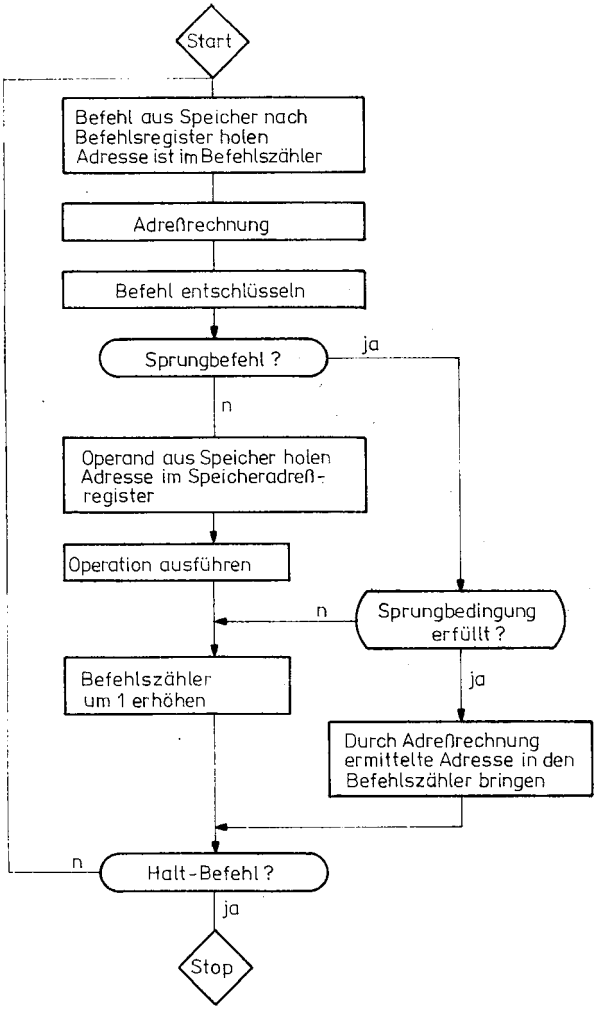


Bild 2.10 Befehlsabarbeitung

cherzelle, aus der der Befehl kommt, steht im Befehlszähler (BZ). Die Ausführung eines Befehls geht in einzelnen Abschnitten, den *Maschinenzyklen*, vor sich. Bild 2.10 zeigt, welche Funktionen bei der Ausführung eines Befehls abgearbeitet werden.

2.6. Ein- und Ausgabesteuerung (E/A-Steuerung)

2.6.1. Prinzip der E/A-Steuerung

Die Ein-/Ausgabe-Steuerung stellt die Verbindung zwischen dem Rechner und den peripheren Geräten her (Bild 2.11).
Die Schnittstellen zwischen Rechner und E/A-Steuerung und zwischen E/A-Steuerung und externem Gerät bestehen aus Daten-, Adreß- und Steuerleitungen. Man spricht vom *Datenbus*, *Adreßbus* und *Steuerbus*. Ist an die E/A-Steuerung nur ein externes Gerät angeschlossen, dann kann der dazugehörige Adreßbus entfallen. Entsprechen die Leitungen einer Schnittstelle einer Normung, so spricht man von einem *Standard-Interface*.

Datenbus
Über den Datenbus werden Dateninformationen übertragen. Er besteht meistens aus 8 (1 Byte) oder 16 Leitungen.

Adreßbus
Der Adreßbus enthält die Adresse der E/A-Steuerung bzw. die Adresse des externen Geräts. Das Gerät entschlüsselt die Adresse

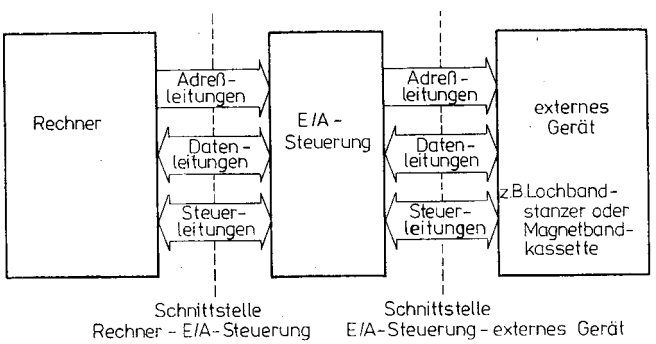


Bild 2.11 Ein-/Ausgabe-Steuerung

und bildet ein Adreßsignal, wenn es durch die richtige Adresse angesprochen wird.

Steuerbus

Der Steuerbus enthält Signale, die den Zeitpunkt der Datenübertragung festlegen. Es sind Melde- und Steuersignale. Meldesignale sind Ausgangssignale eines Geräteteils. Wichtige Meldesignale sind:

- Geräteteil für Daten empfangsbereit;
- Geräteteil hat Daten übernommen;
- Geräteteil hat Daten zur Ausgabe bereitgestellt;
- Fehler im Geräteteil.

Die Meldesignale des sendenden Geräteteils sind gleichzeitig Steuersignale für den empfangenden Geräteteil.

Der Datenaustausch geschieht im allgemeinen nach dem sogenannten *Hand-shake-Prinzip.* (Die anfordernde Stelle gibt ein Anforderungssignal (Request) und wartet, bis die Gegenstelle ein Quittungssignal zurückgibt.)

Beispiel

Die E/A-Steuerung A gibt Daten an das Gerät G (Bild 2.12):

- Die E/A-Steuerung sendet die Adresse des Geräts G und stellt die Daten auf dem Datenbus bereit.
- Die E/A-Steuerung sendet als Zeichen bereitstehender Daten das Signal „A-Bereit".
- Das Gerät G entschlüsselt die Adresse und übernimmt, wenn es bereit ist, die Daten.
- Hat G die Daten übernommen, so sendet es das Signal „Daten übernommen".
- Jetzt kann die E/A-Steuerung die Daten und die Adresse wieder abschalten.

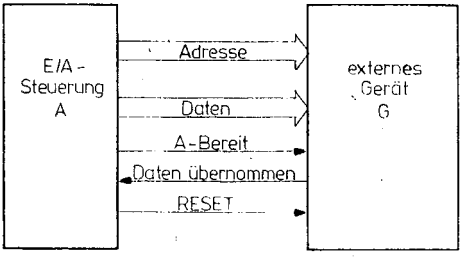


Bild 2.12 Aufbau einer Interface-Schnittstelle

Die Signale „A-Bereit“ und „Daten übernommen“ sind Signale des Steuerbusses. Ein wichtiges Steuersignal ist das *Löschsignal* (RESET), mit dem vom Rechner aus alle angeschlossenen Geräte in die Ausgangsstellung gesetzt werden.

2.6.2. Programmierte Ein- und Ausgabe

Wird ein Rechnerwort vom Rechner über die E/A-Steuerung zum externen Gerät mit Hilfe eines Ein- oder Ausgabebefehls ein- oder ausgegeben, so spricht man von *programmierter Ein-* bzw. *Ausgabe*. Dabei wird meistens vom Datenbus ein Byte in ein Register der CPU eingegeben oder der Inhalt eines Registers auf den Datenbus bereitgestellt. Der Ein- bzw. Ausgabebefehl enthält die Richtung (Ein- oder Ausgabe) und die Geräteadresse. Für die programmierte Ein- und Ausgabe gibt es 4 Befehlsgruppen.

E/A-Befehle Gruppe 1: *(E/A-Steuerbefehle)*	Durch den Befehl werden Funktionssignale zum externen Gerät gesandt, die dort eine bestimmte Funktion auslösen (z. B. Magnetbandkassette starten).
E/A-Befehle Gruppe 2: *(E/A-Testbefehle)*	Durch den Befehl wird der Zustand des externen Geräts abgefragt und in Abhängigkeit vom Zustand ein bedingter Sprung ausgeführt.
E/A-Befehle Gruppe 3: *(Eingabebefehle)*	Durch den Befehl wird die Eingabe eines Datenwortes vom Datenbus in ein spezielles Register der CPU realisiert.
E/A-Befehle Gruppe 4: *(Ausgabebefehle)*	Durch den Befehl wird der Inhalt eines speziellen Registers der CPU auf dem Datenbus bereitgestellt.

2.6.3. Autonome Ein- und Ausgabe

Die *autonome Ein-* und *Ausgabe* ist auch unter den Begriffen DSK (*D*irekter *S*peicher*k*anal) oder DMA (*D*irect *M*emory *A*ccess)

bekannt. Dabei geschieht die Ein- bzw. Ausgabe in Datenblöcken zwischen Speicher und externem Gerät mit Hilfe der E/A-Steuerung (*DMA-Steuerung* genannt).
Die DMA-Steuerung enthält folgende Register:

- Adreßregister zum Adressieren des Speichers;
- Datenregister (Datenpuffer) zur Zwischenspeicherung eines Wortes;
- Geräteadreßregister;
- Blocklängenregister.

Vor dem Auslösen des autonomen Datentransfer werden über die programmierte Ausgabe Adreßregister, Geräteregister und Blocklängenregister mit den entsprechenden Werten gefüllt. Nach dem Start des autonomen Datentransfer geht der weitere Datenaustausch zwischen Speicher und externem Gerät mit Hilfe der DMA-Steuerung vor sich. Zwischen DMA-Steuerung und externem Gerät vollzieht sich der Datenaustausch wie beim programmierten Kanal im Hand-shake-Verfahren. Ist die DMA-Steuerung zum Datenaustausch mit dem Speicher bereit (Datenpuffer voll bei Eingabe, Datenpuffer leer bei Ausgabe), so erfolgt ein Speicherzugriff. Dazu läuft ein Speicherzyklus ab (Speicherzyklus: Adresse an Adreßschlüsselung, Speicheranforderungssignal anlegen, Speicher auf Lesen oder Schreiben schalten, Daten auslesen oder einschreiben). Dieser Speicherzyklus wird bei Rechnern, in denen parallel zur autonomen Ein- und Ausgabe die Arbeit mit der CPU (*C*entral *P*rocessor *U*nit = Rechenwerk und Steuerwerk) möglich ist, eingeschoben. Nach der Ein- bzw. Ausgabe des Datenblocks gibt die DMA-Steuerung ein „Endsignal" ab.

Beispiel
Ausgabe eines Datenblocks vom Speicher. Dabei werden folgende Teilschritte ausgeführt:

- Einstellen des Grundzustands durch E/A-Befehle über den programmierten Kanal;
  a) Füllen des Adreßregisters mit der Anfangsadresse des Datenblocks,
  b) Füllen des Blocklängenregisters mit der Anzahl der Worte,
  c) Füllen des Geräteadreßregisters mit der Adresse des Ausgabegeräts (z. B. Drucker),
  d) Starten zur autonomen Datenausgabe.

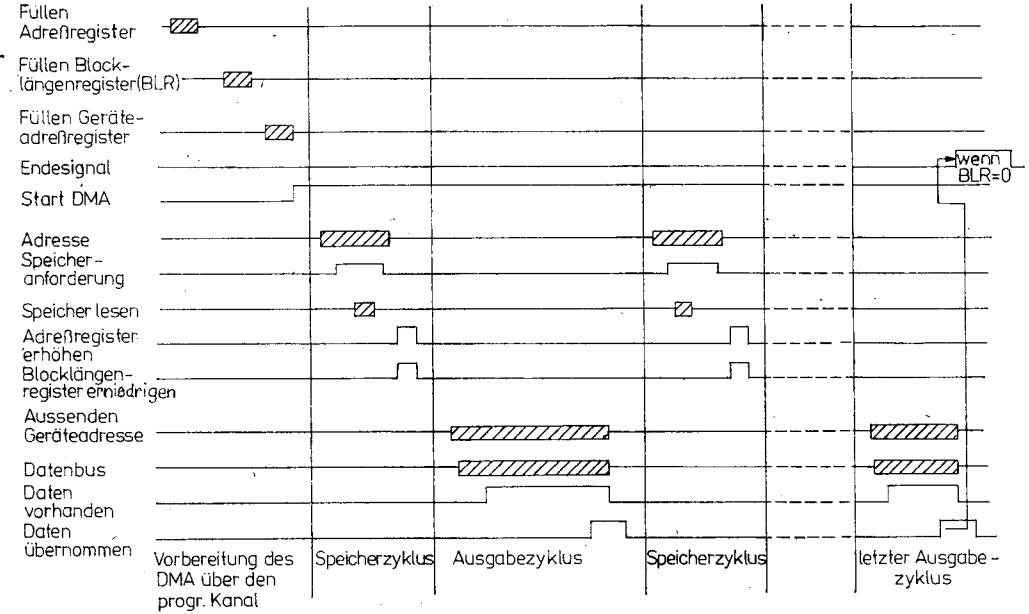


Bild 2.13 Taktdiagramm einer Ausgabe über den DMA-Kanal

- Die DMA-Steuerung fordert einen Speicherzyklus;
  a) Adresse an Adreßentschlüsselung,
  b) Speicheranforderungssignal,
  c) Speicher lesen, Wort nach Datenpuffer,
  d) Inhalt des Adreßregisters um 1 erhöhen,
  e) Inhalt des Blocklängenregisters um 1 erniedrigen.
- Zwischen DMA-Steuerung und Ausgabegerät findet ein Hand-shake-Zyklus statt;
  a) die DMA-Steuerung sendet die Adresse des Ausgabegeräts und stellt die Daten auf dem Datenbus bereit,
  b) die DMA-Steuerung sendet das Signal „Daten vorhanden",
  c) das Ausgabegerät übernimmt das Datenwort,
  d) das Ausgabegerät sendet „Daten übernommen",
  e) die DMA-Steuerung schaltet das Signal „Daten vorhanden" ab.

Speicherzyklus und Hand-shake-Zyklus zum Ausgabegerät werden so lange wiederholt, bis das Blocklängenregister Null ist. Jetzt sendet die DMA-Steuerung das „Endesignal". Bild 2.13 zeigt das Taktdiagramm einer Ausgabe über den DMA-Kanal.

2.7. Programmunterbrechung (INTERRUPT)

Durch eine Programmunterbrechung besteht die Möglichkeit, den Programmablauf durch Signale von außen zu steuern. Ein in den Rechner gegebenes Programmunterbrechungssignal führt zum Abbrechen des gerade laufenden Programms und zur Fortsetzung der Arbeit des Rechners mit einem anderen Programm. Nach einem Programmunterbrechungssignal wird zunächst ein dem Signal zugeordnetes Bedienprogramm abgearbeitet. Erst danach setzt der Rechner das unterbrochene Programm fort. Im einzelnen erzeugt ein Programmunterbrechungssignal folgende Arbeitsgänge:

- Nach Beendigung des gerade laufenden Befehls wird ein INTERRUPT-Zyklus durchlaufen.
- Der Rechner gibt ein Signal „INTERRUPT angenommen" als Quittung nach außen ab.
- Während des INTERRUPT-Zyklus wird ein INTERRUPT-Befehl gebildet und anschließend sofort abgearbeitet.

Der INTERRUPT-Befehl kann ein Sprung in ein Unterprogramm sein. Das Unterprogramm ist das entsprechende Bedienprogramm.

Der Rücksprung ins unterbrochene Programm wird genauso organisiert wie der Rücksprung aus einem Unterprogramm.
Am Anfang des Bedienprogramms bringt man im allgemeinen die Inhalte der Register, die nach dem Rücksprung ins unterbrochene Programm noch benötigt werden, in den *STACK* (Stapelspeicher). Vor dem Rücksprung werden die Register wieder mit den im STACK geretteten Inhalten gefüllt.
Ein Rechner kann einen oder mehrere INTERRUPT-Eingänge haben. Die zu den Eingängen gebildeten INTERRUPT-Befehle unterscheiden sich. Die einzelnen Eingänge lassen sich im allgemeinen durch spezielle Steuerbefehle sperren und öffnen (sie sind maskierbar). Bei mehreren Eingängen besteht eine Vorrangordnung. Kommen an 2 Eingängen die INTERRUPT-Signale gleichzeitig an, so hat das Signal mit der höheren Priorität den Vorrang. Das andere Signal kann vorgemerkt oder ignoriert werden.

2.8. STACK-Organisation

Der STACK (auch Stapelspeicher genannt) ist ein Teil des Arbeitsspeichers, der so organisiert wird, wie man Gegenstände in einem Keller ablagert, d. h., die Informationen werden hineingestapelt. Die Information, die zuletzt hineingebracht wurde, muß man auch als erste wieder herausnehmen. Man sagt, die Speicherung geschieht nach dem Prinzip „last in first out".
Zur Adressierung des STACK dient ein spezielles Register, der STACKPOINTER (Stackzeiger). Für die STACK-Adressierung gelten im allgemeinen folgende Regeln:

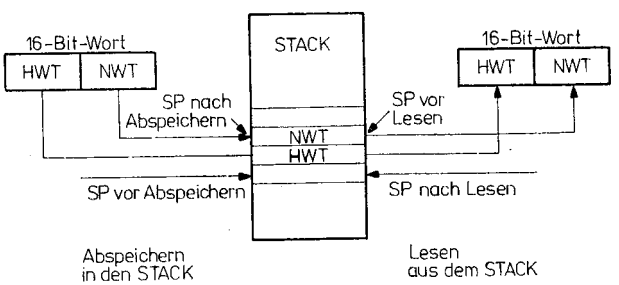


Bild 2.14 Arbeitsweise des STACK-Speichers

- Der STACKPOINTER (SP) zeigt auf das zuletzt eingeschriebene oder auf das nächste zu lesende Byte.
- Der STACK wird von höheren Adressen nach niedrigeren Adressen beschrieben.
- Die Abspeicherung (als PUSH bezeichnet) und das Lesen (als POP bezeichnet) geschieht meistens mit einem 16-Bit-Wort (2 Byte).

Aus Bild 2.14 ist zu ersehen, in welcher Weise ein 16-Bit-Wort in den STACK abgespeichert wird und wie es aus dem STACK gelesen wird, wenn eine Speicherzelle 8 Bit (1 Byte) speichern kann.

2.9. Zusammenstellung der Funktionseinheiten eines Rechners

Aus Bild 2.15 sind die wichtigsten Funktionseinheiten eines Rechners zu ersehen. Für die rein elektronischen Funktionseinheiten (Rechenwerk, Steuerwerk, Speicher, E/A-Steuerung, DMA-Steuerung, INTERRUPT-Steuerung) gibt es spezielle programmierbare Schaltkreise. Dabei werden Rechenwerk und Steuerwerk im Mikroprozessorschaltkreis vereinigt. Für die übrigen Funktionseinheiten gibt es spezielle Schaltkreise.
In Teil 2 und Teil 3 sind Grundlagen und Arbeitsweise der programmierbaren Schaltkreise sowie einige Schaltkreise für die genannten Funktionseinheiten eines Rechners beschrieben.

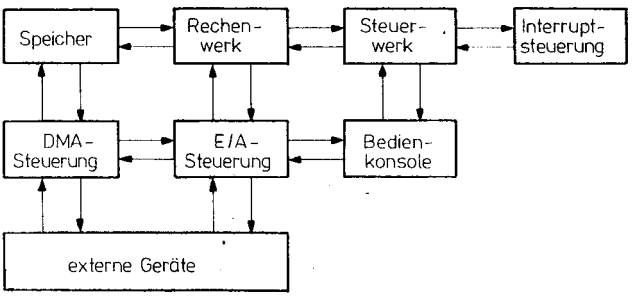


Bild 2.15 Funktionseinheiten eines Rechners

3. Aufbau und Arbeitsweise digitaler Schaltkreise für Mikrorechner

3.1. Übersicht

Die Digitaltechnik befaßt sich mit der Verarbeitung logischer Signale. Durch die Verfeinerung der Technologie war es im Laufe der Zeit möglich, immer mehr der für diese Verarbeitung notwendigen logischen Grundschaltungen auf einem Chip unterzubringen. Von den ersten SSI-Schaltkreisen, in denen einige Transistoren untergebracht sind, bis zu den LSI-Schaltkreisen mit 10^4 bis 10^5 Transistoren ging eine intensive technologische Entwicklung parallel.

Während bei den niedrigintegrierten Schaltkreisen das Spektrum von einfachen Grundschaltungen bis zu speziellen Schaltungen wie Zähler, Register, Decoder, Konverter, Multiplexer, Leitungstreiber reicht, geht die Entwicklung der hochintegrierten Schaltkreise in 2 Richtungen. Die eine Richtung ist der spezielle *Kundenschaltkreis*. Dazu gehören z. B. der Taschenrechnerschaltkreis und der Schaltkreis für Digitaluhren. Diese Schaltkreise realisieren eine spezielle Funktion. Die Schaltung ist fest vorgegeben und für die jeweilige Funktion weitestgehend optimal entwickelt.

Die zweite Richtung sind die *programmierbaren Schaltkreise*. Sie sind Bestandteile eines Rechners. Die Entwicklung verläuft zur Zeit in der Richtung, daß einmal die gesamte Zentraleinheit einschließlich der Ein- und Ausgabekanäle auf einem Chip integriert wird. So entstehen heute bereits Chips, die eine minimale Rechnerkonfiguration (CPU, Speicher, Ein-/Ausgabe-Kanäle) realisieren. Die folgende Zusammenstellung zeigt eine Übersicht existierender Schaltkreistypen.

Digitale Grundschaltkreise
- logische Grundfunktionen
- Flip-Flop-Schaltungen
- Treiber

Analoge Grundschaltkreise
- Operationsverstärker
- NF-Verstärker

- ZF-Verstärker
- Netzregelschaltkreise

MSI-Schaltkreise der Digitaltechnik
- Taktgeneratoren
- Codierer und Decodierer
- Register
- Zähler
- Multiplexer
- Rechenschaltungen
- Speicher
- Bustreiber

MSI-Schaltkreise der Analogtechnik
- Analog-Digital-Umsetzer
- Digital-Analog-Umsetzer

LSI-Kundenschaltkreise
- Taschenrechnerschaltkreise
- Schaltkreise für Digitaluhren

Programmierbare LSI-Schaltkreise
(Schaltkreise der Mikrorechentechnik)
- Mikroprozessoren
- Ein-/Ausgabe-Schaltkreise
- INTERRUPT-Schaltkreise
- Schaltkreise zum direkten Speicherzugriff (DMA-Schaltkreise)

3.2. Schaltalgebra

Eine Reihe von Grundfunktionen, z. B. die Entschlüsselung einer Adresse, müssen meistens durch logische Grundschaltkreise aufgebaut werden. Zum Aufstellen solcher Schaltungen ist die Schaltalgebra eine große Hilfe. Sie ist die Grundlage für das Fällen von Entscheidungen und für das Prüfen von logischen Bedingungen. Mit den Binärziffern 0 und 1 werden sogenannte *logische Operationen* durchgeführt. Zu den logischen Operationen gehören:
die ODER-Funktion bzw. Disjunktion (OR),
die UND-Funktion bzw. Konjunktion (AND),
das EXKLUSIV-ODER (XOR)
und die Nicht-Funktion bzw. NEGATION (NOT).

Die ODER-Funktion

Die ODER-Funktion wird für 2 Binärziffern A und B folgendermaßen definiert:

Wenn A oder B oder beide gleich 1 sind, so ist das Ergebnis gleich 1. Andernfalls ist das Ergebnis gleich 0.

Zur Darstellung der ODER-Funktion wird das Symbol „+" bzw. „$\vee$" (*vel* — lat., oder) genommen.

Es gelten die Kombinationen:

$0 \vee 0 = 0,$
$0 \vee 1 = 1,$
$1 \vee 0 = 1,$
$1 \vee 1 = 1.$

Logische Funktionen werden mit Hilfe einer Funktionstabelle definiert. Die Funktionstabelle enthält die Ausgangssignale, die sich bei den zulässigen Kombinationen der Eingangssignale ergeben. Die Funktionstabelle für die ODER-Funktion ist aus Tabelle 3.1. zu ersehen.

UND-Funktion

Die UND-Funktion wird für 2 Binärziffern A und B folgendermaßen definiert:

Wenn A und B beide gleich 1 sind, so ist das Ergebnis gleich 1. Andernfalls ist das Ergebnis 0.

Zur Darstellung der UND-Funktion wird das Symbol „·" bzw. „$\wedge$" (et — lat., und) genommen. Es ist auch üblich, daß 2 Binärziffern ohne Operationszeichen aneinandergeschrieben werden:
$A \cdot B \triangleq A \wedge B \triangleq AB$

Tabelle 3.1. Funktionstabelle für die ODER-Funktion

Eingänge		Ausgang
A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Tabelle 3.2. Funktionstabelle für die UND-Funktion

Eingänge		Ausgang
A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Es gelten die Kombinationen
$0 \wedge 0 = 0$,
$0 \wedge 1 = 0$,
$1 \wedge 0 = 0$,
$1 \wedge 1 = 1$.
Die Funktionstabelle für die UND-Funktion ist in Tabelle 3.2. dargestellt.

Das EXKLUSIV-ODER
Beim EXKLUSIV-ODER ist das Ergebnis der Verknüpfung von A und B gleich 1, wenn die Eingangssignale verschieden sind, und 0, wenn die Eingangssignale gleich sind.
Für das EXKLUSIV-ODER wird das Symbol $\oplus$ bzw. das Symbol $\veebar$ verwendet. Es gelten folgende Kombinationen:
$0 \oplus 0 = 0$,
$0 \oplus 1 = 1$,
$1 \oplus 0 = 1$,
$1 \oplus 1 = 0$.

Tabelle 3.3. Funktionstabelle für das EXKLUSIV-ODER

Eingänge		Ausgang
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Die Funktionstabelle für das EXKLUSIV-ODER ist aus Tabelle 3.3. zu ersehen.

Die Nicht-Funktion (NEGATION)

Die Nicht-Funktion bezieht sich nur auf eine Binärziffer. Sie wird durch einen Querstrich über die betreffende Ziffer dargestellt. ($\overline{A}$ heißt NEGATION von A.)

Bei der NEGATION ist das Ergebnis 1, wenn A gleich 0 ist, und 0, wenn A gleich 1 ist.

Es gilt:

$\overline{1} = 0,$

$\overline{0} = 1.$

Tabelle 3.4. zeigt die Funktionstabelle für die NEGATION.

Rechenregeln für logische Operationen

a) Die Negation einer negierten Größe ergibt die Größe selbst:
   (1) $\overline{\overline{a}} = a.$

b) Für die logischen Operationen mit 2 Größen gelten folgende Vertauschungsgesetze (Kommutativgesetz):
   (2) $a \vee b = b \vee a,$
   (3) $a \wedge b = b \wedge a,$
   (4) $a \oplus b = b \oplus a.$

Insbesondere gilt:

$1 \vee a = 1,$
$1 \wedge a = a,$
$0 \vee a = a,$
$0 \wedge a = 0,$
$a \vee \overline{a} = 1,$
$a \wedge \overline{a} = 0.$

Tabelle 3.4. Funktionstabelle für die NEGATION

Eingang	Ausgang
A	$\overline{A}$
0	1
1	0

c) Für zusammengefaßte Ausdrücke gilt das Distributivgesetz:

(5) $(a \vee b) \vee c = a \vee (b \vee c)$,

(6) $(a \wedge b) \wedge c = a \wedge (b \wedge c)$,

(7) $a \wedge b \vee c = (a \vee c) \wedge (b \vee c)$.

Weiterhin gilt:

(8) $\overline{a \wedge b} = \bar{a} \vee \bar{b}$,

(9) $\overline{a \vee b} = \bar{a} \wedge \bar{b}$,

(10) $\bar{a} \vee ab = \bar{a} \vee b$,

(11) $a \vee \bar{a}b = a \vee b$.

Die Regeln (1) bis (11) lassen sich überprüfen, wenn man jeweils für die rechte und linke Seite die Funktionstabelle aufstellt und dabei die Funktionstabellen für die UND-Funktion, die ODER-Funktion, die NEGATION und für das EXKLUSIV-ODER verwendet.

Mit Hilfe der genannten Umformungsregeln kann man logische Ausdrücke wesentlich vereinfachen. Bei der technischen Realisierung lassen sich Schaltungen optimieren und damit Bausteine einsparen.

Beispiel

Der logische Ausdruck

$T = ad \vee a\bar{c} \vee bd \vee bc \vee \bar{a}b \vee \bar{a}\bar{b}$

läßt sich wie folgt vereinfachen:

$T = ad \vee a\bar{c} \vee bd \vee bc \vee \bar{a}$,

$T = d \vee \bar{a} \vee bc \vee \bar{c}$,

$T = d \vee \bar{a} \vee b \vee \bar{c}$.

Ein sehr häufig vorkommendes Problem ist die Aufstellung eines

Tabelle 3.5. Funktionstabelle für die Addition von 3 Dualziffern (2 Summanden a, b und Übertrag u)

a	b	u	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

logischen Ausdrucks bei vorgegebener Funktionstabelle. Will man z. B. die Summe von 3 Dualziffern bilden (2 Operanden a und b und einen Übertrag u vom niedrigen Stellenwert), so kann die Funktionstabelle 3.5. aufgestellt werden. Die Summe s ist 1, wenn eine Ziffer 1 oder alle 3 Ziffern 1 sind. Sind alle 3 Ziffern 1, so tritt ein Übertrag in die nächste Stelle auf. Sind 2 Ziffern 1, dann ist die Summe 0 und der Übertrag in die nächste Stelle 1.

s ist hier eine logische Funktion von den 3 Größen a, b und u. Aus Tabelle 3.5. kann man ablesen, daß $s = 1$ ist, wenn

$a = 0; b = 0; u = 1$

oder

$a = 0; b = 1; u = 0$

oder

$a = 1; b = 0; u = 0$

oder

$a = 1; b = 1; u = 1.$

Statt $a = 0$ kann man auch sagen, es muß $\bar{a} = 1$ sein, d. h., es ist $s = 1$, wenn

$\bar{a} = 1; \bar{b} = 1; u = 1$ bzw. $\bar{a}\,\bar{b}\,u = 1$

oder

$\bar{a} = 1; b = 1; \bar{u} = 1$ bzw. $\bar{a}\,b\,\bar{u} = 1$

oder

$a = 1; \bar{b} = 1; \bar{u} = 1$ bzw. $a\,\bar{b}\,\bar{u} = 1$

oder

$a = 1; b = 1; u = 1$ bzw. $a\,b\,u = 1.$

Den Ausdruck $\bar{a}\,\bar{b}\,u$ nennt man die Elementarkonjunktion P_1 (a, b, u). Sie ist nur für die Belegung 0 0 1 ($a = 0; b = 0; u = 1$) gleich 1, sonst 0. Ebenso ist die Elementarkonjunktion P_2 (a, b, u) $= \bar{a}b\bar{u}$ nur für die Belegung 0 1 0 ($a = 0; b = 1; u = 0$) gleich 1, sonst 0.

Die ODER-Funktion der Elementarkonjunktion $\bar{a}b\bar{u}$, $\bar{a}bu$, $ab\bar{u}$ und abu ist damit gerade dort 1, wo s nach Tabelle 3.5. gleich 1 ist. Sonst ist diese ODER-Funktion 0. Danach läßt sich schreiben:

$s = \bar{a}\,\bar{b}\,u \vee \bar{a}\,b\,\bar{u} \vee a\,\bar{b}\,\bar{u} \vee a\,b\,u.$

Durch Anwendung des Distributivgesetzes (7) erhält man:

$s = u\,(\bar{a}\bar{b} \vee ab) \vee \bar{u}\,(\bar{a}b \vee a\bar{b})\,.$

Tabelle 3.6. Tabelle der Elementarkonjunktionen Pj

	a_n	a_{n-1}	a_{n-2} ...	a_2	a_1	Elementarkonjunktionen
Belegung 0	0	0	0 ... 0	0		$P_0 = \bar{a}_n \bar{a}_{n-1} \bar{a}_{n-2} \dots \bar{a}_2 \bar{a}_1$
Belegung 1	0	0	0 ... 0	1		$P_1 = \bar{a}_n \bar{a}_{n-1} \bar{a}_{n-2} \dots \bar{a}_2 a_1$
Belegung 2	0	0	0 ... 1	0		$P_2 = \bar{a}_n \bar{a}_{n-1} \bar{a}_{n-2} \dots a_2 \bar{a}_1$
.			.			.
.			.			.
.			.			.
.			.			.
Belegung 2^n-2	1	1	1 ... 1	0		$P_{2^n-1} = a_n a_{n-1} a_{n-2} \dots a_2 \bar{a}_1$
Belegung 2^n-1	1	1	1 ... 1	1		$P_{2^n} = a_n a_{n-1} a_{n-2} \dots a_2 a_1$

Mathematische Zusammenfassung

Liegt eine Funktionstabelle mit n Größen $a_1 a_2 \dots a_n$ vor und ist T die Ergebnisfunktion, so bilde man zunächst alle Elementarkonjunktionen P_j, für deren Belegung j T = 1 ist (Tabelle 3.6.). Die ODER-Funktion dieser Elementarkonjunktion P_j ist die logische Funktion T:

$$T = \bigvee_{j=0}^{2^n-1} Q_j P_j .$$

Dabei ist $Q_j = \begin{cases} 0, \text{ wenn T für die Belegung } j \text{ gleich 0 ist;} \\ 1, \text{ wenn T für die Belegung } j \text{ gleich 1 ist.} \end{cases}$

Durch Vereinfachung des Ausdrucks für T mit Hilfe der Rechenregeln für logische Funktionen erhält man die gewünschte Funktion.

Beispiel

Ein Baustein soll mit der Adresse 14 ausgewählt werden. Die Gesamtadresse ist 16 Bit lang. Das Auswahlsignal CS (chip-select) ist nur dann aktiv (= 1), wenn auf den Adreßleitungen eine 14 steht:

Adreß-
ltgn.: A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0
Adres-
se 14: 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
CS =

$$P_{14} = \bar{A}_{15}\,\bar{A}_{14}\,\bar{A}_{13}\,\bar{A}_{12}\,\bar{A}_{11}\,\bar{A}_{10}\,A_9\,\bar{A}_8\,\bar{A}_7\,A_6\,\bar{A}_5\,A_4\,\bar{A}_3\,\bar{A}_2\,\bar{A}_1\,\bar{A}_0.$$

3.3. Logische Grundschaltkreise

Die Verarbeitung von Binärsignalen basiert auf wenigen Grundschaltungen. Die wichtigsten Grundschaltungen sind die Realisierung des logischen UND, des logischen ODER und der NEGATION. Diese Funktionen werden in den meisten Schaltkreisreihen durch NOR- oder NAND-Schaltungen gebildet. Der Leser informiere sich an dieser Stelle ausführlicher über die Schaltkreisserie *D 10*, z. B. in der Reihe *electronica* (Hefte 141, 155, 156).

Eine sehr wichtige Eigenschaft in der Mikroelektronik ist die Möglichkeit, Schaltkreise ausgangsseitig zusammenzuschalten. Diese Eigenschaft haben nicht alle Schaltkreise. Eine Form der Zusammenschaltung ist der Wired-OR-verdrahtete Ausgang mit offenem Kollektor (Bild 3.1). Dabei erhalten Kollektorausgänge aller Schaltkreise einen gemeinsamen Arbeitswiderstand. Der Nachteil dieser Schaltung besteht im hohen Stromverbrauch, da der Arbeitswiderstand um so kleiner sein muß, je mehr Schaltkreise ausgangsseitig parallelgeschaltet werden.

Eine zweite Möglichkeit der ausgangsseitigen Parallelschaltung bietet die Tri-state-Schaltung. Bei dieser Schaltung kann der Ausgang neben den Pegeln H und L noch einen dritten hochohmigen Zustand einnehmen. Der hochohmige Zustand wird erreicht, wenn sowohl der obere als auch der untere Ausgangstransistor sperren (Tiefpegel am Steuereingang $\bar{X}$; Bild 3.2).

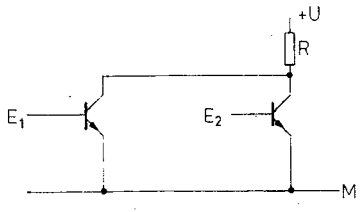


Bild 3.1
Parallelschaltung von Ausgängen nach dem Wired-OR-Prinzip

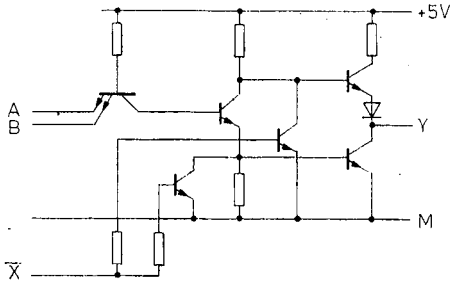


Bild 3.2 Erzeugung des hochohmigen Zustands am Ausgang Y eines Schaltkreises durch Sperren beider Ausgangstransistoren (L-Pegel am Steuereingang $\overline{X}$)

Zum Beispiel läßt sich der Baustein *D 150* in Tri-state-Schaltung betreiben, da über $\overline{X}$ ein Sperren des oberen Ausgangstransistors möglich ist, wenn $\overline{X}$ L-Pegel führt. Bild 3.3 zeigt die Ansteuerung des Bausteins *D 150* für den Tri-state-Betrieb. Für die Umschaltung in den hochohmigen Zustand ist der Steueranschluß E_S vorgesehen. Um den unteren Ausgangstransistor zu sperren, werden alle zweiten AND-Eingänge (B, D) auf L-Pegel gelegt. Damit ist der untere Ausgangstransistor gesperrt. Der obere Ausgangstransistor ist durch L-Pegel an $\overline{X}$ gesperrt.

3.4. Informationsspeicherung

3.4.1. Überblick

Die Grundeinheit der Information ist das *Bit*; die Grundschaltung zur Speicherung eines Bit das *Flip-Flop*. Werden mehrere Bit zusammengefaßt, so ergibt sich ein Bit-Muster. Das Bit-Muster

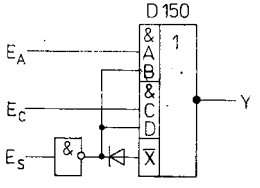


Bild 3.3
Tri-state-Steuerung am Baustein *D 150*

kann 1 Byte (8 Bit) oder Wort sein. Ein Byte oder Wort wird im *Register* gespeichert (zwischengespeichert). Zur Speicherung vieler Worte (den Daten eines Rechners) ist eine große Anzahl von Registern erforderlich. Sie bilden den *Speicher* eines Rechners.
Flip-Flop-Schaltungen sind die Grundlage aller Halbleiterspeicher. Über Aufbau und Arbeitsweise von Flip-Flop-Schaltungen informiere sich der Leser in [1]. Sie sollen an dieser Stelle nicht näher behandelt werden.

3.4.2. Register

Ein Register dient zur Speicherung eines Bit-Musters. Es besteht deshalb aus einer entsprechenden Anzahl von Flip-Flop. Je nach der zusätzlichen Funktion, die das Register noch zu erfüllen hat, werden dazu RS-Flip-Flop, D-Flip-Flop oder JK-Flip-Flop gewählt.
Registerschaltungen können auch einfache Operationsschritte ausführen, z. B. die gespeicherte Information um eine oder mehrere Stellen nach rechts oder links verschieben (Schieberegister). Bild 3.4 zeigt ein einfaches Speicherregister aus RS-Flip-Flop für 4 Bit. Wenn das Torungssignal STR (Strobe) den Pegel H hat, wird die Information E_3 E_2 E_1 E_0 im Register gespeichert. H-Pegel des Signals RESET löscht den Registerinhalt. Der Registerinhalt läßt sich an A_3 A_2 A_1 A_0 abnehmen.
Bild 3.5 zeigt ein 4-Bit-Schieberegister aus JK-Flip-Flop. Die einzelnen Stellen des Registers können von außen parallel statisch oder dynamisch mit Hilfe der Eingänge E_0 bis E_3 gesetzt werden. Das ermöglicht eine parallele Eingabe. Über die Ausgänge A_0, $\bar{A}_0$ bis A_3, $\bar{A}_3$ ist eine parallele Ausgabe der Stellen des Registers möglich. Ein 4-Bit-Muster, das im Register steht, kann seriell über den Ausgang A, $\bar{A}$ ausgegeben werden. Der Schiebeprozeß wird durch den Takt T veranlaßt. Durch die mit jedem Takt ausgelöste Rechtsverschiebung können auch über den Eingang E, $\bar{E}$ neue Bit-Muster seriell eingegeben werden. Das Register ist somit für die Umsetzung parallel → seriell und seriell → parallel geeignet.
Mit Hilfe eines Dynamikvorsatzes (Kondensatorspeicher) läßt sich ein Schieberegister auch mit einem RS-Flip-Flop aufbauen.

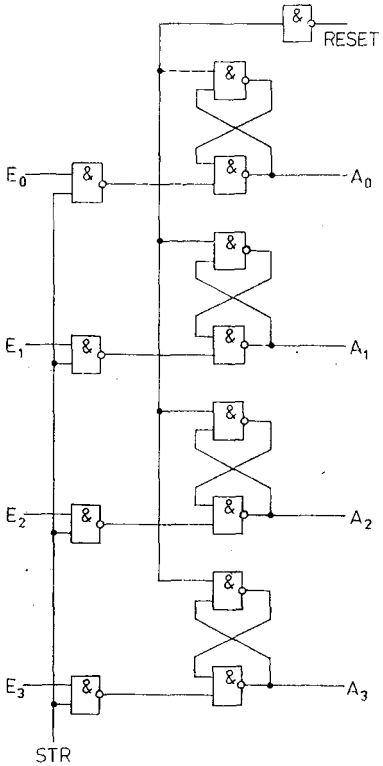


Bild 3.4
Speicherregister für
4 Bit aus RS-Flip-Flop

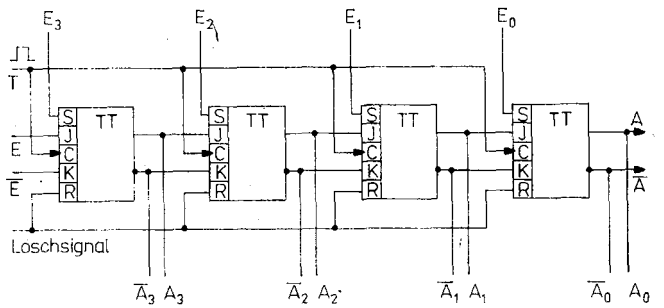


Bild 3.5 4-Bit-Schieberegister aus JK-Flip-Flop

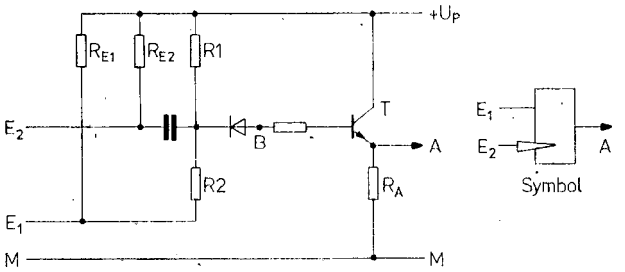


Bild 3.6 Dynamikvorsatz (Kondensatorspeicher) zur Realisierung von Schieberegistern mit RS-Flip-Flop

Bild 3.6 zeigt die Schaltung eines solchen Dynamikvorsatzes. Mit E_1 wird die Schaltung vorbereitet. Hat E_1 L-Potential, so steht an Punkt B eine kleine positive Spannung. Ein negativer Impuls bei E_2 sperrt den Transistor T kurzzeitig und erzeugt am Ausgang A einen L-Impuls. Hat E_1 H-Potential, so liegt bei B die volle Spannung U_p an. Ein negativer Impuls an E_2 bringt Transistor T nicht bis zur Sperrschwelle. Der Transistor T sperrt im Fall $E_1 = L$ auch, wenn zum gleichen Zeitpunkt, zu dem der

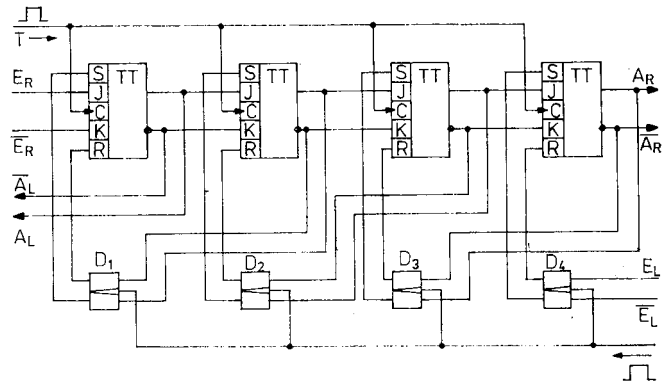


Bild 3.7 Zwei-Richtungs-Schieberegister mit Dynamikvorsatz für die Linksverschiebung

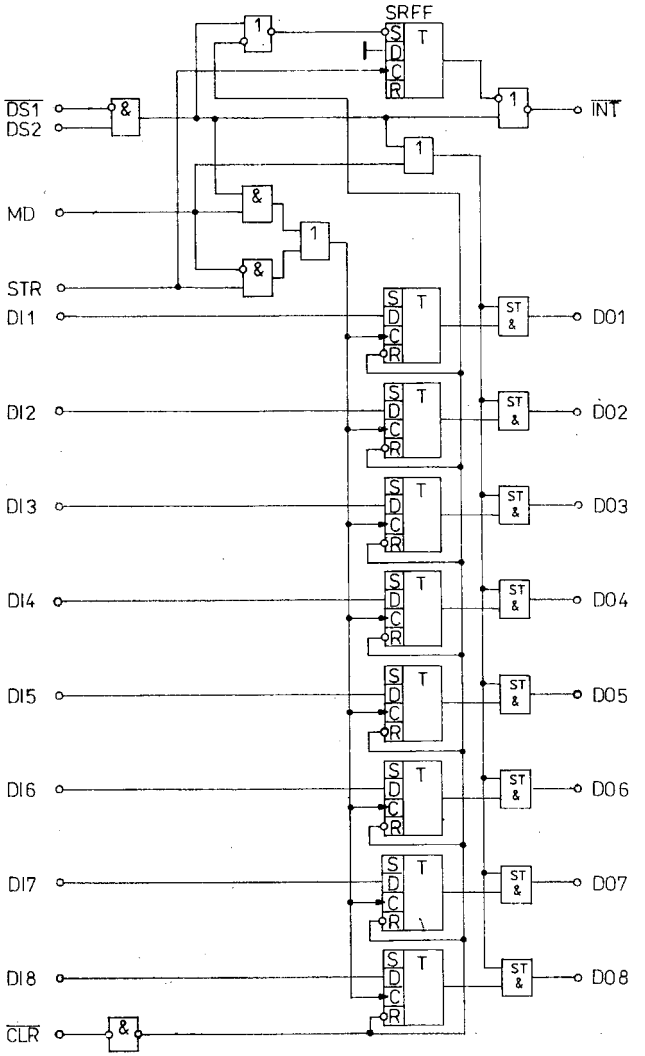


Bild 3.8 Schaltbild des Bausteins *8212*

negative Impuls an E_2 erscheint, E_1 H-Pegel erhält, da das Potential bei B erst langsam über R2 umgeladen wird.
Bild 3.7 zeigt ein Zwei-Richtungs-Schieberegister für 4 Bit mit Kondensatorspeicher für die Linksverschiebung. Die Dynamikvorsätze D_1 bis D_4 ermöglichen ein kurzzeitiges Speichern der Information bei der Linksverschiebung.
Für den Aufbau von Mikrorechnern gibt es Schaltkreise, in die komplette Register integriert sind. Bild 3.8 zeigt die Schaltung des Schaltkreises *8212*, der zur Dateneingabe und -ausgabe verwendet werden kann. Er besteht aus einem 8-Bit-Register mit D-Flip-Flop und einer Steuerschaltung.

Arbeitsweise des Bausteins 8212

Der Schaltkreis *8212* arbeitet in 2 Betriebsarten, die durch MD (Mode) gesteuert werden.
Betriebsart 1: Eingang MD hat H-Pegel
Die Information der einzelnen Registerstellen kann direkt an den Ausgängen DO1 bis DO8 abgenommen werden. Wenn $\overline{DS1}$ L-Pegel und DS2 H-Pegel hat, dann wird die Information von den Eingängen DI1 bis DI8 in die D-Flip-Flop eingegeben.
Betriebsart 2: Eingang MD hat L-Pegel
Mit H-Pegel an STR (Strobe) wird die Information in die D-Flip-Flop gespeichert.
Wenn $\overline{DS1}$ L-Pegel und DS2 H-Pegel führt, so kann die Information, die im Register gespeichert ist, an den Ausgängen DO1 bis DO8 abgenommen werden.
Im Fall 2 setzt außerdem das Signal STR mit der HL-Flanke das Flip-Flop SRFF zurück und damit den $\overline{INT}$-Ausgang auf L-Pegel. Durch $\overline{DS1} \cdot DS2$ wird das Flip-Flop SRFF wieder gesetzt. Die Leitung $\overline{INT}$ dient später als INTERRUPT-Anforderung für Mikroprozessorschaltkreise (s. Teil 2, Abschnitt 2.6. und 3.6.).
Durch L-Pegel am Eingang $\overline{CLR}$ (Löschen) werden die D-Flip-Flop des Registers rückgesetzt und das Flip-Flop SRFF gesetzt ($\overline{INT}$ = H, d. h. nicht aktiv). Die Ausgänge DO1 bis DO8 sind mit einer Tri-state-Steuerung versehen, d. h., wenn die Ausgangsleitungen gesperrt sind, ist ihr Ausgang hochohmig.

Zusammenstellung der Funktionen des Bausteins 8212

MD = H: $DE \cdot \overline{DS1} \cdot DS2 \rightarrow$ Registerinhalt
$DA =$ Registerinhalt

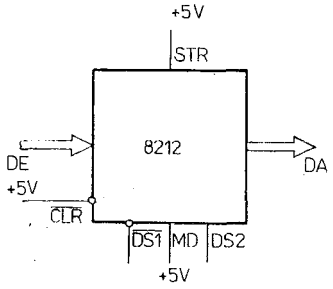


Bild 3.9
Der Baustein *8212* als Datenpuffer

MD = L: DE · STR → Registerinhalt, DA = Registerinhalt
· $\overline{DS1}$ · DS2
STR (H → T) → $\overline{SRFF}$ → INTERRUPT-Anforderung

$\overline{CLR}$ = L: Löschen Register und Setzen SRFF

Das INT-Signal ist aktiv ($\overline{INT}$ = L), wenn
INT = $\overline{DS1}$ · DS2 ∨ $\overline{SRFF}$ = 1 ist.

Anwendung des Bausteins 8212

Der Baustein *8212* läßt sich sehr universell für die Bussteuerung und Datenpufferung anwenden. Bild 3.9 zeigt den Einsatz des Bausteins als Datenpuffer. Die Eingabe in das Register geschieht mit $\overline{DS1}$ · DS2 = H. Die Ausgabe aus dem Register ist jederzeit möglich.

Bild 3.10 zeigt die Verschaltung des Bausteins *8212* als Eingabepuffer. Die Eingabe in das Register läßt sich über das Signal STR vornehmen. Das Signal STR aktiviert gleichzeitig das INT-Signal. Damit wird angezeigt, daß der Datenpuffer voll ist.

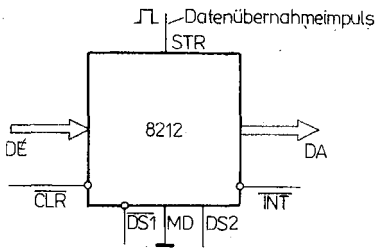


Bild 3.10
Der Baustein *8212* als Eingabetor

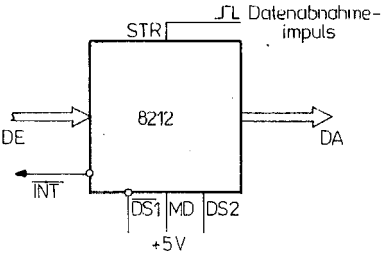


Bild 3.11
Der Baustein *8212* im Hand-shake-Prinzip

Die Ausgabe beginnt in dem Moment, wo $\overline{DS1} \cdot DS2 = H$ ist. Mit diesem Signal wird gleichzeitig das SRFF-Flip-Flop gesetzt. Das INT-Signal verschwindet jedoch erst mit der HL-Flanke von $\overline{DS1} \cdot DS2$.

Bild 3.11 zeigt die Verschaltung des Bausteins *8212* im Hand-shake-Prinzip für einen Ausgabepuffer. Die Eingabe ins Register geschieht mit dem Signal $\overline{DS1} \cdot DS2 = H$. Der Inhalt des Re-

Tabelle 3. 7. Zusammenstellung einiger Registerschaltkreise

Typ		Kurzcharakteristik
MH 7475	ČSSR	4-Bit-D-Register mit Enable
74174 PC	UVR	6-Bit-D-Flip-Flop, löschbar
74175 PC	UVR	4-Bit-D-Flip-Flop, löschbar
74259 PC	UVR	8-Bit-Register, adressierbar
K 500 TM 131		4-Bit-D-Register (ECL)
D 191 C/D	DDR	8-Bit-Schieberegister
D 195 C/D	DDR	4-Bit-rechts/links-Schieberegister
MH 7496	ČSSR	5-Bit-Schieberegister, E/A parallel mit Löschung
MH 74164		8-Bit-Schieberegister Eingabe seriell; Ausgabe parallel
74165 PC	UVR	8-Bit-Schieberegister
74166 PC		Eingabe seriell; Ausgabe parallel
K 155 IR 13	UdSSR	8-Bit-rechts/links-Schieberegister
74194 PC	UVR	4-Bit-rechts/links-Schieberegister
74195 PC	UVR	4-Bit-Schieberegister Eingabe parallel; Ausgabe parallel

gisters liegt ständig an DA. Den zur Abnahme des Datenwortes verwendeten Strobeimpuls führt man gleichzeitig an den Strobeeingang, wodurch das INT-Signal aktiv wird (als Rückmeldung, daß die Daten abgeholt wurden).
In Tabelle 3.7. sind einige Registerschaltkreise zusammengestellt.

3.4.3. Speicher

Während ein Register dazu dient, 1 Bit-Muster zu speichern, hat der Speicher die Aufgabe, eine größere Anzahl Bit-Muster (Worte genannt) aufzunehmen. Im Laufe der Entwicklung der Rechentechnik wurden die unterschiedlichsten Speicherprinzipien entwickelt (Magnettrommelspeicher, Laufzeitspeicher, Speicherbildröhre, Ringkernspeicher, Halbleiterspeicher). In der Mikrorechentechnik wird ausschließlich der Halbleiterspeicher verwendet. Deshalb soll auch nur auf ihn eingegangen werden.
Speicherschaltkreise kann man nach der Art der Anwendung oder nach der Herstellungstechnologie einteilen. Nach der Art der Anwendung gibt es

ROM (*R*ead *O*nly *M*emory — Nur-Lesespeicher),
PROM (*P*rogrammable *R*ead *O*nly *M*emory — Programmierbarer ROM),
EPROM (*E*rasable *PROM* — Löschbarer und programmierbarer ROM),
RAM (*R*andom *A*ccess *M*emory — Schreib-Lese-Speicher).

Nach der Herstellungstechnologie unterscheidet man Speicherschaltkreise in
TTL-Technik,
p-MOS-Technik,
n-MOS-Technik,
CMOS-Technik,
I^2L-Technik.
Eine Beschreibung der unterschiedlichen Techniken findet der Leser in [2].

RAM (Schreib-Lese-Speicher)
Statische Schreib-Lese-Speicher
Bei den statischen RAM bilden 2 rückgekoppelte NICHT-Gatter die Speicherzellen, wobei den beiden Zuständen die Information 0 oder 1 zugeordnet wird.

Aus technologischer Sicht gibt es solche Speicher in TTL-Technik und MOS-Technik. Speicher in TTL-Technik sind sehr schnell, sie haben aber einen großen Leistungs- und Platzbedarf. MOS-Bausteine erfordern weniger Platz. Die Schaltkreise in p-Kanal-MOS-Technik sind sehr langsam und benötigen mehrere Betriebsspannungen. Heute werden MOS-Schaltkreise fast ausschließlich in n-Kanal-Technik ausgeführt. Moderne MOS-Schaltungen sind voll TTL-kompatibel. Bei der CMOS-Technik bestehen wesentlich günstigere Betriebsspannungen (3 bis 15 V). Ihre Leistungsaufnahme liegt wesentlich unter der der TTL-Technik. Bild 3.12 zeigt den Aufbau einer Speicherzelle in TTL-Technik, Bild 3.13 den Aufbau einer Speichermatrix mit 16 Speicherzellen.

Um einen Speicherplatz innerhalb der Speichermatrix anzusteuern, müssen die X- und Y-Leitung H-Signal führen. Damit wird die Speicherzelle aktiviert, bei der X- und Y-Leitung gleichzeitig H-Signal haben. In der adressierten Speicherzelle fließt der Kollektorstrom des leitenden Transistors in die entsprechende Schreib-Lese-Leitung L oder H. An den Ausgängen der Schreib-Lese-Leitungen befinden sich Leseverstärker (LV), die aus dem Kollektorstrom ein logisches L (Q_L) oder H (Q_H) bilden.

Zum Einschreiben einer Information wird nach der Adressierung der Speicherzelle auf eine der Schreib-Lese-Leitungen ein H-Signal gegeben. Zur Speicherung einer 0 wird auf die L-Leitung ein H-Signal (Eingang E_L) und zur Speicherung einer 1 auf die H-Leitung (Eingang E_H) ein H-Signal gegeben.

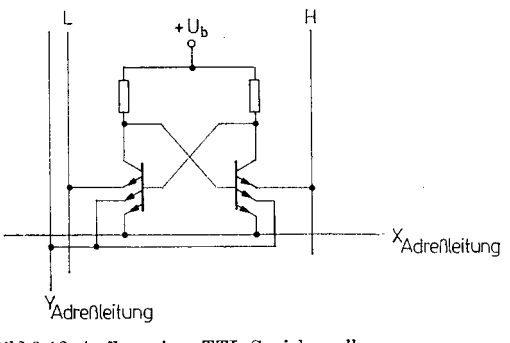


Bild 3.12 Aufbau einer TTL-Speicherzelle

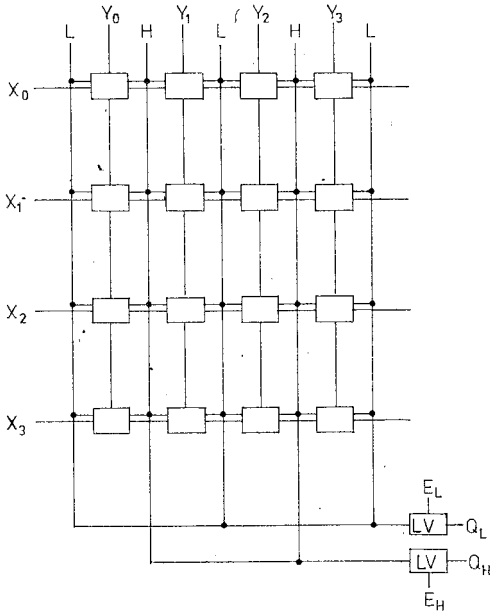


Bild 3.13 TTL-Speichermatrix mit 16 Speicherzellen

Die in Bild 3.13 dargestellte Anordnung der Flip-Flop ist in den TTL-Bausteinen *7481* und *7484* realisiert. Es kann immer nur 1 Bit gelesen bzw. eingeschrieben werden. Um eine 0 einzuschreiben, muß man den E_L-Eingang ansteuern, zum Einschreiben einer 1 den E_H-Eingang. Die gelesene Information liegt an Q_L bzw. Q_H.

Bild 3.14 zeigt den Baustein *7489*. Er hat eine 16×4-Struktur, d. h. eine Speicherkapazität von 16 Wörtern zu je 4 Bit. Der Baustein *7489* ist ein voll decodiertes RAM.

Schreibvorgang: Wenn die Freigabeeingänge $\overline{CS}$ (chip-select, Bausteinauswahl) und $\overline{WR}$ (Write) auf L-Signal liegen, wird die an den Dateneingängen D_1 bis D_4 bereitgestellte Information in die durch die Adreßeingänge A_0 bis A_3 angesteuerten Speicherzellen gebracht. Die Y-Leitungen tragen dabei die negierte Information, die auch gleichzeitig an $\overline{Q}_1$ bis $\overline{Q}_4$ anliegt. Die Flip-

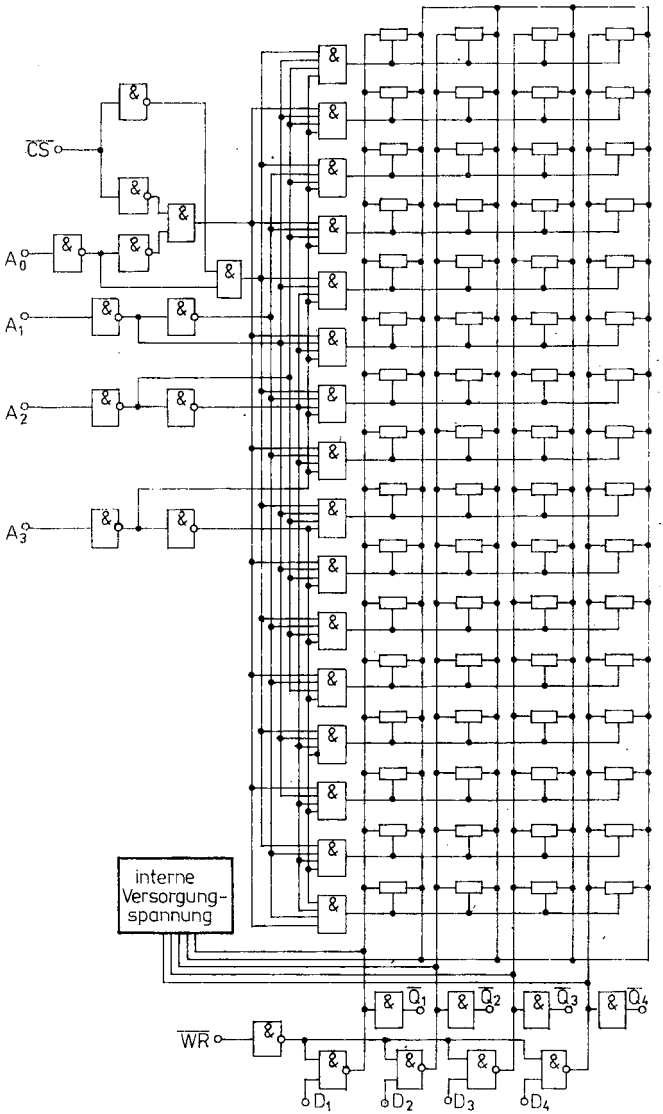


Bild 3.14 Logischer Aufbau des Speicherbausteins *7489*

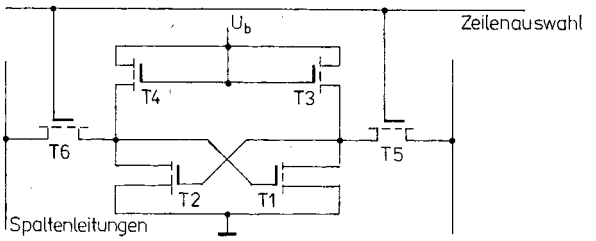


Bild 3.15 Aufbau einer MOS-Speicherzelle

Flop schalten mit L-Signal auf den Y-Leitungen und mit H-Signal auf den X-Leitungen.

Lesevorgang: Wenn $\overline{CS}$ L-Signal und $\overline{WR}$ H-Signal erhält, so liegt an den Ausgängen $\overline{Q_1}$ bis $\overline{Q_4}$ der negierte Inhalt der durch A_0 bis A_3 adressierten Speicherzelle.

Statische Speicher in MOS-Technik

Bild 3.15 zeigt die Prinzipschaltung einer MOS-Speicherzelle für statische RAM. Die Lastwiderstände sind wegen des geringen Platzbedarfs durch Transistoren T_3, T_4 ersetzt.

Lesevorgang: Wenn die Transistoren T_5 und T_6 über die Zeilenauswahl durchgesteuert sind, dann kann die Information als Signalunterschied an den Spaltenleitungen abgenommen werden.

Schreibvorgang: Leiten T_5 und T_6, so läßt sich das Flip-Flop über die Spaltenleitungen in den gewünschten Zustand setzen.

Bild 3.16 zeigt das Prinzip eines Halbleiterspeichers mit 256 Flip-Flop aus p-Kanal-MOS-Transistoren. Die Flip-Flop sind in einer Ebene so angeordnet, daß jederzeit über 16 X- und Y-Koordinaten ein direkter Zugriff zu jedem Flip-Flop besteht. Wenn an den Koordinaten X_m, Y_m das Potential $-U_{DD}$ anliegt, werden die Transistoren T_5 bis T_8 durchgesteuert.

Zur Speicherung einer „1" liegen z. B. an SL_2 18 V über T_9 (T_9 ist geöffnet) und an SL_1 0 V über R_3 und R_4 (T_{10} ist gesperrt). Dadurch leitet T_3, und T_4 sperrt. Die durch T_1 und T_2 gebildeten Lastwiderstände halten diese Stellung auch nach Abtrennen der Adresse. Zur Speicherung einer „0" haben SL_2 0 V und SL_1 18 V Potential. Beim Lesen sind T_9 und T_{10} geöffnet. Die Widerstände R_1, R_2 und R_3, R_4 liegen zu den Lastwiderständen T_1 und T_2 parallel. Durch R_2 bzw. R_4 fließt Strom entsprechend den Po-

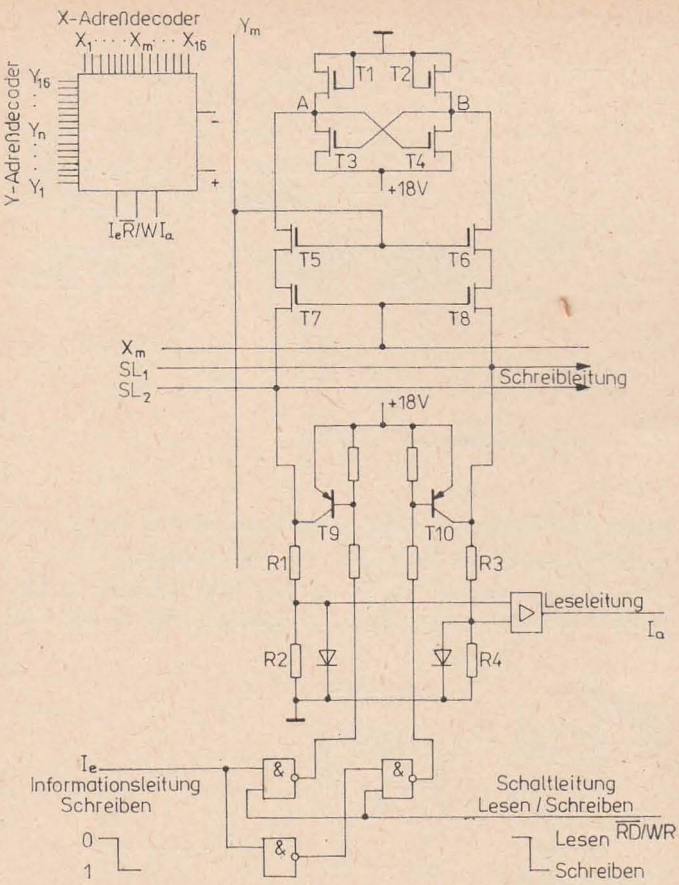


Bild 3.16 Halbleiterspeicher mit 256 Flip-Flop aus MOS-p- Kanal-Transistoren (negierte Logik)

tentialen bei A und B. Der Spannungsabfall über R_2 und R_4 wird über einen Differenzverstärker als Informationssignal abgenommen.

In weiterentwickelten Speicherschaltkreisen sind die Adreßdecodierung sowie eine Bausteinauswahllogik enthalten.

Bild 3.17 zeigt den Aufbau eines 256 × 1 volldecodierten sta-

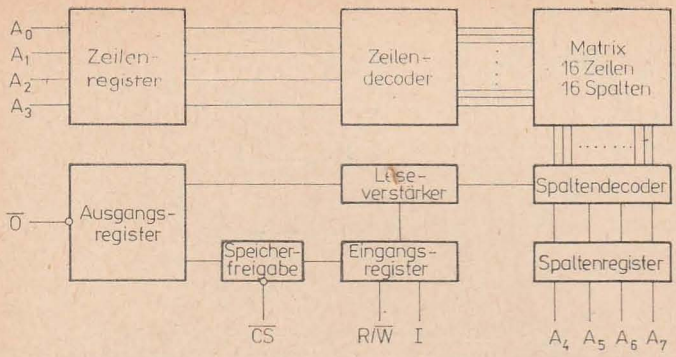


Bild 3.17 MOS-RAM mit 256 × 1 Zellen

tischen MOS-RAM. Der Baustein wird mit $\overline{CS} = L$ angesteuert. Die Zeilenauswahl geschieht über die Adreßeingänge A_0 bis A_3, die Spaltenauswahl über A_4 bis A_7. Hat $R/\overline{W}$ L-Signal, so wird die an I liegende Information eingeschrieben. Bei H-Signal an

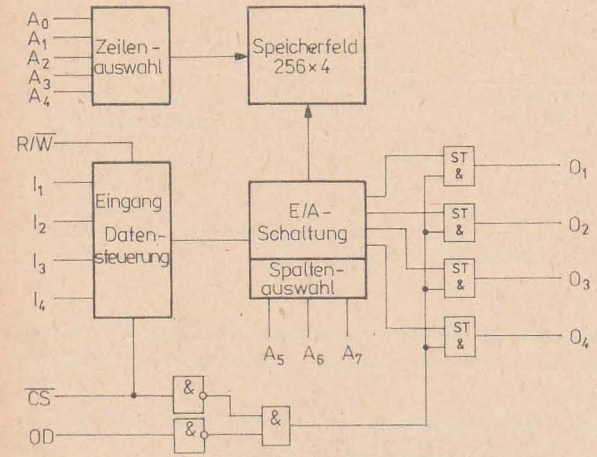


Bild 3.18 MOS-RAM mit 256 × 4 Zellen

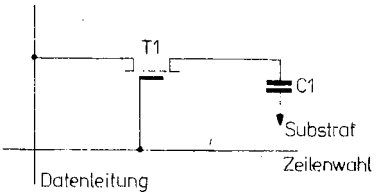


Bild 3.19
Aufbau einer dynamischen RAM-Speicherzelle mit einem Transistor

$R/\overline{W}$ erscheint an $\overline{O}$ der invertierte Inhalt der ausgewählten Speicherzelle. Mit H-Signal an $\overline{CS}$ ist der Speicher blockiert.
Bild 3.18 zeigt den Aufbau eines 256 × 4-RAM. Über A_0 bis A_4 werden die Zeilen und über A_5 bis A_7 die Spalten ausgewählt. Bei L-Signal an $R/\overline{W}$ lassen sich die Daten auslesen. Die Information erscheint bei O_1 bis O_4. Bei H-Signal an $R/\overline{W}$ werden die an I_1 bis I_4 liegenden Informationen in die adressierte Speicherzelle eingeschrieben. Liegt an OD H-Signal, so sind die Ausgänge $\overline{O}_1$ bis $\overline{O}_4$ hochohmig. $\overline{CS}$ dient zur Chipwahl. Mit L-Signal an $\overline{CS}$ arbeitet der Speicherbaustein.

Dynamische Schreib-Lese-Speicher
Bei den dynamischen RAM wird die Information durch Aufladung eines Kondensators gespeichert. Bild 3.19 zeigt eine Ein-Transistor-Speicherzelle. Sobald T_1 leitet, erscheint die Information auf der Datenleitung. Wegen der auftretenden Leckströme muß die Kondensatorladung periodisch (ungefähr alle 2 ms) regeneriert (wieder aufgeladen) werden.

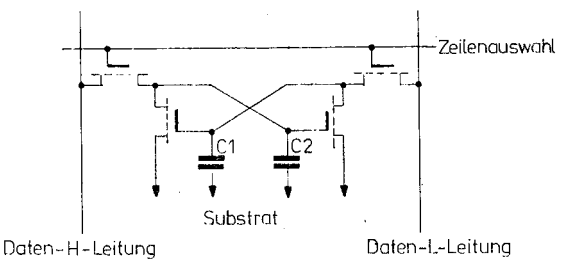


Bild 3.20 Dynamische RAM-Speicherzelle nach dem Flip-Flop-Prinzip

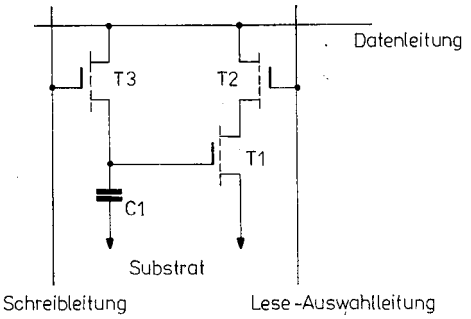


Bild 3.21 Dynamische RAM-Speicherzelle

Die ersten dynamischen RAM waren den statischen RAM angelehnt. Bild 3.20 zeigt den Aufbau einer solchen Speicherzelle. Aus Bild 3.21 ist der heutzutage meistens verwendete Aufbau einer dynamischen RAM-Zelle zu ersehen.
Zum Einschreiben muß die Schreibleitung H-Signal haben und die Datenleitung die Information (L- oder H-Signal) tragen. Über T_3 wird C_1 entsprechend aufgeladen.
Zum Lesen muß die Lese-Auswahlleitung H-Signal führen (T_2 leitet). Hat C_1 H-Signal, so wird T_1 leitend, und auf der Daten-

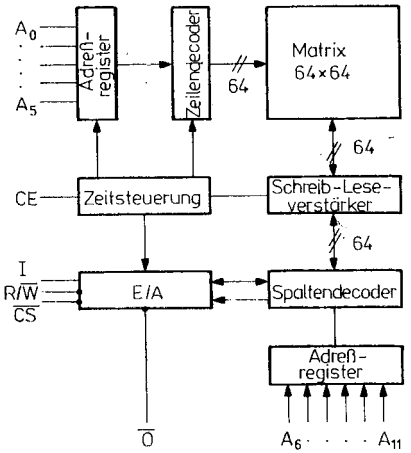


Bild 3.22
Dynamischer
4 K × 1-RAM

leitung entsteht L-Potential. Hat C_1 L-Potential, dann bleibt der Weg T_2-T_1-Substrat hochohmig, und die Datenleitung erhält H-Potential. Auf der Datenleitung entsteht das negierte Informationssignal der Speicherzelle. Lese-Auswahlleitung und Schreibleitung bestimmen meistens die Zeilenauswahl, die Datenleitung bestimmt die Spaltenauswahl. Aus Bild 3.22 ist der Aufbau eines dynamischen 4 K × 1-RAM zu ersehen.

In Bild 3.22 bedeuten:

A_0 bis A_{11} Adreßeingänge;

I — Dateneingang

$R/\overline{W}$-Schreibfreigabe ($R/\overline{W}$ = L≙ Schreiben; $R/\overline{W}$ = H ≙ Lesen);

$\overline{O}$ — Datenausgang;

$\overline{CS}$ — Bausteinauswahl ($\overline{CS}$ = L ≙ Freigabe);

CE— Speicherfreigabe (chip enable).

Jede einzelne Zeile muß innerhalb von 2 ms einen Auffrischzyklus erhalten. Während dieser Zeit darf nicht neu eingeschrieben werden. Das ist möglich bei

einem Lesezyklus: $R/\overline{W}$ = H, $\overline{CS}$ beliebig;

einem Schreibzyklus: $R/\overline{W}$ = L, dann muß $\overline{CS}$ H-Potential haben. Der Eingang CE schaltet den Speicher im nichtaktiven Zustand (L-Pegel an CE) auf geringen Leistungsverbrauch. Er beträgt im nichtaktiven Zustand ≈ 1,3 mW, zusätzlich ≈ 7 mW für das Auffrischen.

Nur-Lese-Speicher (ROM)

Ein ROM (*R*ead *O*nly *M*emory) ist ein vom Hersteller programmierbarer Festwertspeicher. Der Inhalt eines programmierten ROM läßt sich nicht mehr ändern. Der so hergestellte Dateninhalt kann nur ausgelesen werden.

Bild 3.23 zeigt den Aufbau eines Masken-ROM. Die Basiselektroden, die die Verbindung zur Zeilenauswahl herstellen, werden durch Wegätzen einer SiO_2-Schicht, die zwischen einem Metallband (Zeilenleitungen) und dem Substrat liegt, gebildet. Dazu verwendet man eine Maske, die an jeder Programmierstelle ein Programmierungsloch hat.

Die Ausgabe erfolgt über die Spaltenleitungen. Für jeden Ausgang existiert ein Leseverstärker, der nur arbeiten kann, wenn am CE-Eingang H-Signal anliegt. Damit sperren die 4 unteren Tran-

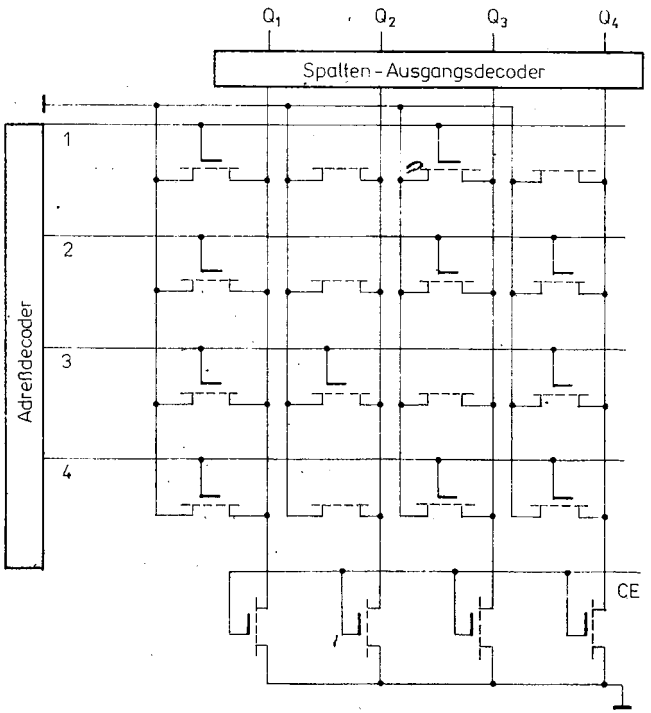


Bild 3.23 Aufbau eines Masken-ROM

sistoren (p-Kanal-MOS-Transistoren), und es kann kein Strom gegen Masse fließen. Soll Zeile 2 angewählt werden, dann muß die Zeilenleitung L-Signal bekommen. Die MOS-Transistoren, deren Basiselektrode vorhanden ist, bilden jetzt eine leitende Verbindung zur Masse. Der fließende Strom wird in den Leseverstärkern in ein logisches Signal umgewandelt. In Spalte 1, 3 und 4 kann über Zeile 2 ein solcher Strom fließen. Die Leseverstärker von Q_1, Q_3 und Q_4 erkennen deshalb ein L-Signal, der Leseverstärker Q_2 dagegen ein H-Signal.

Programmierbare ROM (PROM)

PROM können vom Anwender mit Hilfe eines Programmier-

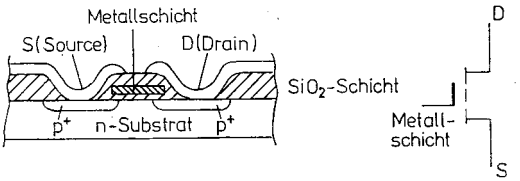


Bild 3.24 Aufbau einer MOS-PROM-Speicherzelle

geräts programmiert werden. Die Programmierung geschieht mit elektrischen Impulsen.

MOS-PROM

Bild 3.24 zeigt den Aufbau einer MOS-PROM-Speicherzelle. In einem n-Substrat liegen die beiden p-Zonen. Bei dem Programmiervorgang wird der Drainanschluß der Speicherzelle mit Masse verbunden und der Source-Anschluß an eine Spannungsquelle angeschlossen. Bei niedriger Spannung fließt durch die Speicherzelle nur ein kleiner Strom. Bei höherer Spannung wird der Strom größer. Er verläßt die p-Zone der Source und fließt über die Metallschicht. Dabei kommt es zu einer Wanderung von Metallatomen. Durch diese Wanderung entsteht zwischen den p-Zonen und der Metallschicht eine elektrisch leitende Verbindung. Der Programmierungsstrom beträgt etwa 150 mA.

TTL-Dioden-PROM

Bild 3.25 zeigt eine Diodenmatrix. Im unprogrammierten Zustand sind die Dioden bereits vorhanden, aber nicht angeschlossen. Erst durch Wegätzen einer SiO_2-Schicht werden die Katoden der Dioden mit den Y-Leitungen elektrisch verbunden.

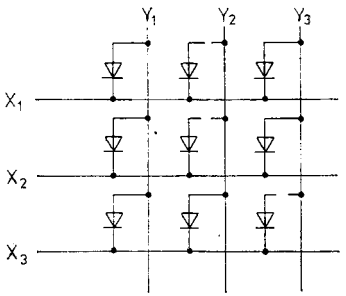


Bild 3.25
PROM-Dioden-Matrix

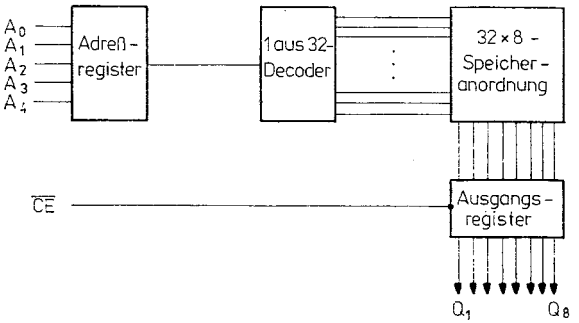


Bild 3.26 32 × 8-TTL-Dioden-PROM

An den Y-Achsen ist durch die Leseverstärker an jeder Leitung ein H-Signal vorhanden. Durch die Adressierung wird auf eine X-Achse ein L-Signal gelegt. Über die angeschlossenen Dioden kann ein Strom gegen Masse fließen. Die Leseverstärker wandeln diesen Strom in das logische Signal um.
Bild 3.26 zeigt den Aufbau eines 32 × 8-TTL-Dioden-PROM. Eine andere Möglichkeit zur Herstellung von TTL-Dioden-PROM ist das Fusable-link-Verfahren. Bild 3.27 zeigt den Aufbau eines FL-PROM. Im unprogrammierten Zustand sind zwischen den Dioden und den Y-Leitungen Brücken in Form eines Sicherungsdrahts (fusable-link). Die Speicherzelle wird durch Abschmelzen dieser Verbindung mit Hilfe eines Stromimpulses programmiert.
Ein anderes Verfahren zur Herstellung von programmierbaren Festwertspeichern ist das AIM-Verfahren (*A*valanche-*I*nduced-*M*igration). Die Matrix besteht aus einem Transistorfeld (Bild

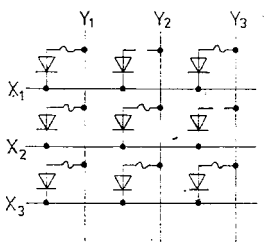


Bild 3.27
Aufbau eines PROM nach dem Fusable-link-Verfahren

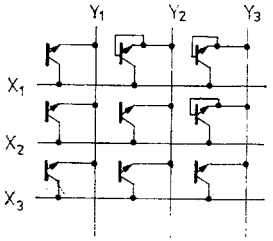


Bild 3.28
Aufbau eines PROM nach dem AIM-Verfahren

3.28). Bei der unprogrammierten Speicherzelle ist die Basis nicht angeschlossen. Beim Programmieren wird die Basis-Emitter-Diode der zu programmierenden Speicherzelle mit Stromstößen abgebaut. Sie hat im unprogrammierten Zustand einen Widerstand von etwa 100 kΩ, im programmierten Zustand von etwa 200 Ω (Bild 3.29). Die Stromstöße zur Programmierung betragen 200 mA bei einer Spannung von 32 V und 7,5 μs Dauer. Diese Stromstöße werden so lange wiederholt, bis der Widerstand der Speicherzelle unter 200 Ω liegt.

Löschbare Festwertspeicher
(EPROM — *E*rasable *PROM*, REPROM — *Re*versible PROM).
Für das Löschen verwendet man ultraviolettes Licht. Dabei wird der gesamte Speicherinhalt gelöscht. Nach dem Löschen läßt sich der Speicher neu programmieren.
Bild 3.30 zeigt den Aufbau einer löschbaren Speicherzelle. Ansteuerungsgate und Floating-Gate sind Poly-Siliziumschichten. Das Ansteuerungsgate wird mit den Zeilenleitungen verbunden. Das Floating-Gate dient zur Ladungsspeicherung. Beim Pro-

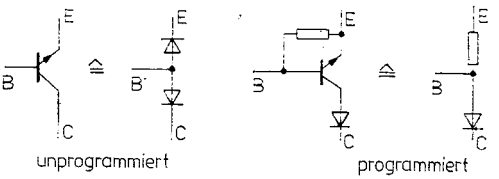


Bild 3.29 Programmierung einer PROM-AIM-Zelle

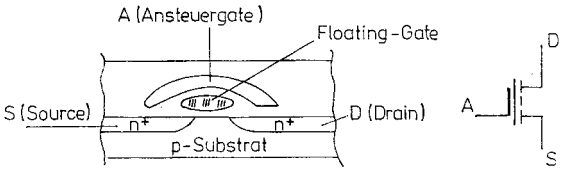


Bild 3.30 Aufbau einer mit UV-Licht löschbaren PROM-Speicherzelle

grammieren wird über die Spaltenleitungen durch Injektion energiereicher Elektronen das Floating-Gate aufgeladen. Die gespeicherte Ladung verändert die Schwellwertspannung der Zelle. Bild 3.31 zeigt den logischen Aufbau eines EPROM.

Zur Programmierung wird $\overline{CS}$ auf +12 V gelegt. An die Adreßeingänge A_0 bis A_9 wird wie beim Lesen die Wortadresse angelegt. Die Programminformation kommt an die Datenleitung D_0 bis D_7. Nachdem Adressen und Daten eingestellt sind, wird ein Programmierungsimpuls je Adresse an den PROGR-Eingang gegeben. Das einmalige Durchlaufen aller Adressen bezeichnet man als *Programmierschleife*. Die Anzahl der Programmierschleifen ist von der Impulsdauer des PROGR-Signals abhängig.

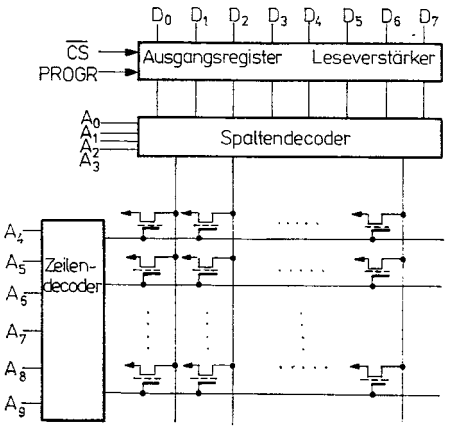


Bild 3.31 Aufbau eines EPROM

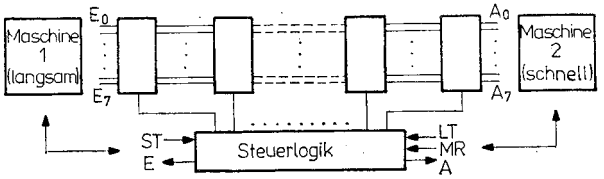


Bild 3.32 Pufferspeicherprinzip

Das Löschen geschieht mit UV-Licht über vorhandene Quarzfenster. Bei einigen EPROM wird eine Wellenlänge von 25,37 μm (2537 Å) bei einer Dosis von 10 Ws/cm^{-2} und einer Löschzeit von 20 bis 30 min angegeben. Dem Strahler darf kein Filter vorgesetzt sein. Durch die UV-Strahlung werden die Elektronen vom Floating-Gate gelöst.
Außer mit UV-Licht löschbare PROM gibt es noch PROM, die sich mit einer hohen Spannung löschen lassen (VEPROM — *V*oltage *e*rasable *PROM*). Diese Bausteine haben anstelle des Quarzfensters eine Metallplatte mit einem Anschlußpin. Durch Anlegen einer hohen Spannung von etwa 10 min Dauer mit umgekehrten Vorzeichen gegenüber der Spannung beim Programmieren werden die Elektronen aus dem Floating-Gate herausgezogen.

Pufferspeicher
Eine spezielle Speicherlogik stellt der Pufferspeicher nach dem Prinzip „first in, first out" dar. Mit ihm lassen sich zwei mit unterschiedlicher Geschwindigkeit arbeitende Maschinen anpassen. Bild 3.32 zeigt die grundsätzliche Struktur eines solchen Speichers. Die Daten werden über E_0 bis E_7 mit Hilfe des Schreibtaktes ST eingeschrieben. Durch das Signal E meldet der Speicher, daß er leere Zellen hat. Wenn der Speicher gefüllt ist, wird das durch das Signal A gemeldet. Ist das Signal A aktiv, so können die Daten mit Hilfe des Lesetaktes LT ausgelesen werden. Das Signal A wird inaktiv, wenn der Speicher leer ist oder wenn Maschine 2 ein Abschlußsignal MR (*m*aster *r*eset) abgibt. Durch MR wird der Speicher gelöscht, A inaktiv und E aktiv. Dann kann ein neues Einschreiben beginnen. Mit jedem Signal ST und LT wird der Inhalt des Speichers um 1 Zelle nach rechts geschoben.

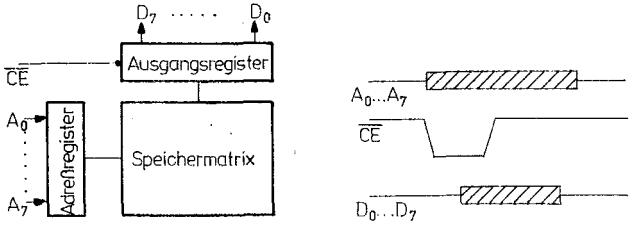


Bild 3.33 ROM-Speicherschaltkreis *U 501 D*

3.4.4. Zusammenstellung einiger Speicherschaltkreise

ROM-Schaltkreise

U 501 D

ROM-Speicherschaltkreis mit 256 × 8-Organisation (Bild 3.33)

Der Speicher hat einen Chip-enable-Eingang ($\overline{CE}$). Bei L-Signal an $\overline{CE}$ gibt der Baustein den Inhalt der an den Adreßleitungen A_0 bis A_7 adressierten Speicherzelle an D_0 bis D_7 ab.

8308

ROM-Schaltkreis mit 1024 × 8-Organisation (Bild 3.34)

Der Schaltkreis hat 2 Eingänge, $\overline{CS_1}$ und CS_2, zur Bausteinauswahl. Mit $\overline{CS_1} \cdot CS_2 = H$ wird die durch A_0 bis A_9 adressierte Speicherzelle gelesen. Ihr Inhalt erscheint an D_0 bis D_7.

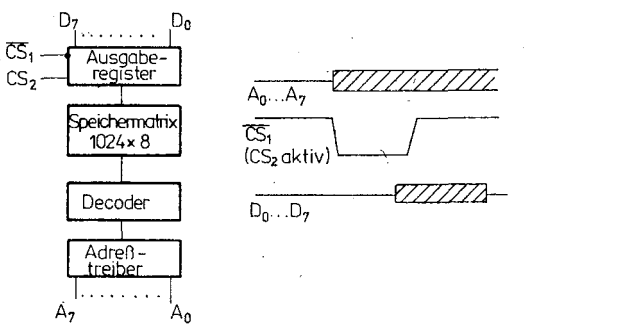


Bild 3.34 ROM-Speicherschaltkreis *8308*

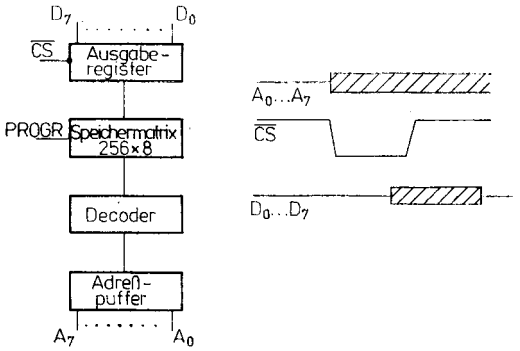


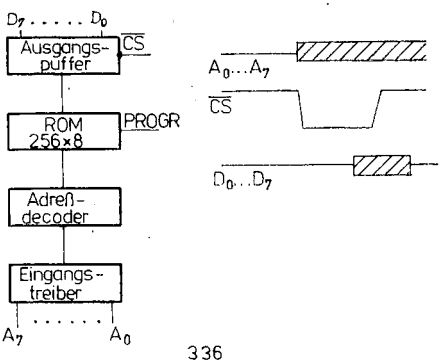
Bild 3.35 EPROM-Speicherschaltkreis *8702 A*

8702 A

EPROM-Schaltkreis mit 256 × 8-Organisation (Bild 3.35)

Der Schaltkreis *8702 A* ist ein löschbarer und elektrisch programmierbarer ROM. Der Speicher hat einen Eingang $\overline{CS}$ (chip-select). Wenn an $\overline{CS}$ L-Potential liegt, wird der Inhalt der durch die Adreßleitung A_0 bis A_7 adressierten Speicherzelle an D_0 bis D_7 bereitgestellt.

Der Schaltkreis wird durch Anlegen eines Programmimpulses am Eingang PROGR programmiert. Dabei erhalten die Adreßleitungen und Stromversorgungsleitungen impulsförmig höhere Spannungen als im Arbeitsbetrieb.



336

Bild 3.36 EPROM-Speicherschaltkreis *1702 A*

Tabelle 3.8. Anschlüsse des Bausteins 1702 A beim Lesen und Programmieren

Anschluß	Lesen	Programmieren
1–3, 17–21	Adreß-Bits	
4–11	Daten-Bits	
12	5 V	0 V
13	5 V	Progr.-Imp. — 48 V
14	0 V	0 V
15	5 V	12 V
16	— 9 V	— 35 V ... — 40 V
22, 23	5 V	0 V
24	— 9 V	— 9 V

Das Löschen geschieht mit UV-Licht. Dazu muß der Baustein etwa 10 bis 20 min mit UV-Licht der Wellenlänge 25,37 nm (2537 Å) mit einer Intensität von 6 Ws · cm^{-2} bestrahlt werden.

1702 A

EPROM-Schaltkreis mit 256 × 8-Organisation (Bild 3.36)

Der Schaltkreis *1702 A* ist ein löschbarer und elektrisch programmierbarer ROM. Der Speicher hat einen Eingang $\overline{CS}$ (chip-select). Wenn an $\overline{CS}$ L-Potential liegt, dann wird der Inhalt der durch die Adreßleitung A_0 bis A_7 adressierten Speicherzelle an D_0 bis D_7 bereitgestellt.

Zum Programmieren wird an den Eingang PROGR ein Impuls gelegt. Vor diesem Programmierimpuls muß an die Adreßleitung die Adresse der zu programmierenden Speicherzelle und an die Datenleitungen das Programmwort gelegt werden.

Aus Tabelle 3.8. sind die Anschlüsse des Bausteins *1702 A* beim Lesen und Programmieren zu ersehen. Das Löschen geschieht mit UV-Licht der Wellenlänge 25,37 μm (2537 Å).

8708

EPROM-Schaltkreis mit 1024 × 8-Organisation (Bild 3.37)

Der Schaltkreis *8708* ist ein löschbarer und elektrisch programmierbarer ROM. Das Lesen geschieht in der Weise, daß der Ein-

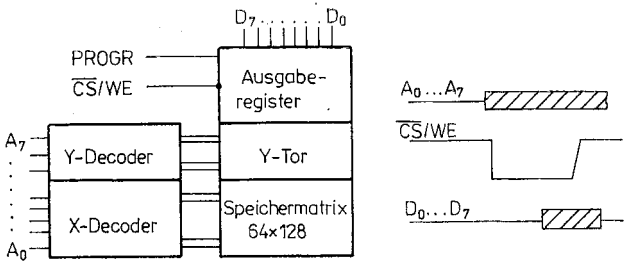


Bild 3.37 EPROM-Speicherschaltkreis *8708*

gang $\overline{CS}$/WE auf L-Potential gelegt und an die Adreßleitung A_0 bis A_9 die Adresse der gewünschten Speicherzelle gegeben wird. An D_0 bis D_7 kann der Inhalt der Speicherzelle abgenommen werden.

Zum Programmieren des Schaltkreises wird an den Eingang $\overline{CS}$/WE eine Spannung von 12 V und an den Eingang PROGR ein Impuls von 26 V gelegt. Den Speicherschaltkreis löscht man mit UV-Licht der Wellenlänge 25,37 μm (2537 Å) mit einer Intensität von 10 Ws · cm^{-2} in einer Zeitdauer von 20 bis 30 min.

RAM-Schaltkreise

2102

RAM-Schaltkreis mit 1024 × 1-Organisation (Bild 3.38)

Der Schaltkreis hat einen Bausteinauswahleingang $\overline{CS}$. Bei $\overline{CS}$ = L ist der Baustein angesteuert. Durch den Anschluß R/$\overline{W}$ läßt sich der Schaltkreis zwischen Lesen und Schreiben um-

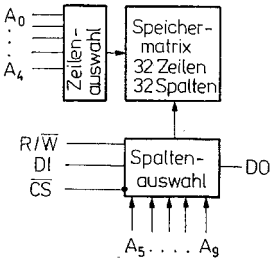


Bild 3.38
RAM-Speicherschaltkreis *2102*

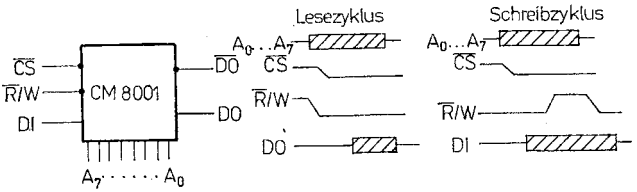


Bild 3.39 Statischer RAM-Speicherschaltkreis *CM 8001*

schalten. Beim Schreiben ($R/\overline{W} = L$) wird die an DI anliegende Information gespeichert. Bei Lesen ($R/\overline{W} = H$) ist die gespeicherte Information an DO abnehmbar.

CM 8001

Statischer RAM-Schaltkreis mit 256 × 1-Organisation (Bild 3.39)

Das Lesen des Schaltkreises geschieht dadurch, daß an den Eingang $\overline{CS}$ L-Potential gelegt wird. Die Information erscheint am Ausgang DO und in negierter Form an $\overline{DO}$.

Zum Beschreiben des Bausteins muß $R/\overline{W} = H$ sein und die Information an den Eingang DI gelegt werden.

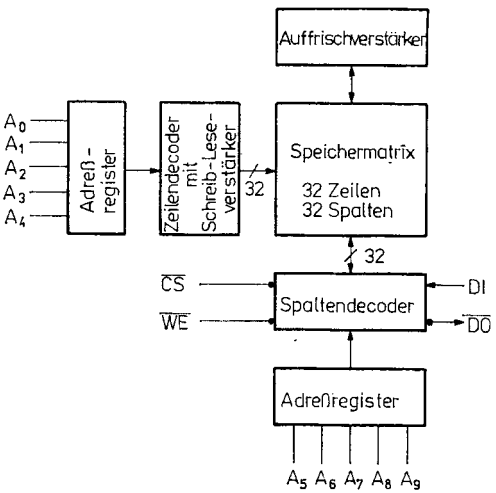


Bild 3.40 Dynamischer RAM-Speicherschaltkreis *U 253 D*

U 253 D

Dynamischer RAM-Schaltkreis mit 1024 × 1-Organisation (Bild 3.40)

Der Schaltkreis *U 253 D* ist ein dynamischer Schreib-Lese-Speicher. Zum Lesen muß der Eingang $\overline{CS}$ L-Potential haben und die Adresse an A_0 bis A_9 gelegt werden. Der Eingang $\overline{WE}$ bleibt auf H-Potential. Die gelesene Information liegt negiert am Ausgang $\overline{DO}$ an. Zum Einschreiben ($\overline{WE}$ = L) legt man die Information am Eingang DI an.

Zum Auffrischen wird die ausgewählte Zelle gelesen und gleichzeitig wieder eingeschrieben. Dazu müssen alle 32 Zeilen der Speichermatrix, die von den Adressen A_0 bis A_4 angesteuert werden, innerhalb von 2 ms mindestens einmal gelesen worden sein. Die Belegung der Adressen A_5 bis A_9 (Spalten) ist dabei ohne Bedeutung.

Tabelle 3.9. Zusammenstellung einiger Speicherschaltkreise

Typ		Kurzcharakteristik
MSI-Schaltkreise		
D 181 C/D	DDR	16×1-RAM
K 155 RU 3	UdSSR	16×1-RAM
K 155 RU 2	UdSSR	16×4-RAM
MH 74188	ČSSR	32×8-PROM
MH 74 S 201	ČSSR	256×1-RAM mit Tri-state-Ausgang
74200 PC	UVR	256×1-RAM mit Tri-state-Ausgang
K 155 RP 1	UdSSR	4×4-Bit-Register-Stapel
LSI-Schaltkreise		
2716		2K×8-UV-EPROM (nur 5 V)
2758		1K×8-UV-EPROM (nur 5 V)
2708		1K×8-UV-EPROM (+12 V, + 5 V, − 5 V)
2704		512×8-UV-EPROM (+ 12 V, + 5 V, − 5 V)
8101		256×4-RAM (+ 5 V)
8111		256×4-RAM (+ 5 V)
2112		256×4-RAM (+ 5 V)
K 565 PY 1 A		dynamischer 4K×1-RAM (− 5 V, + 5 V, + 12 V)

Weitere Speicherschaltkreise sind in Tabelle 3.9. zusammengestellt.

3.5. Codier- und Decodierschaltungen

Umcodierungen werden beim Aufbau von Mikrorechnern sehr häufig gebraucht. Beispiele sind die Realisierung der Anzeige durch LED-Elemente, die Zuordnung von Ziffern- und Funktionstasten zu den entsprechenden Zahlendarstellungen und die Entschlüsselung von dualen Adressen. Für viele dieser Funktionen gibt es integrierte Schaltkreise. Stehen keine speziellen Schaltkreise zur Verfügung, so lassen sich entsprechende Schaltungen auch mit logischen Grundschaltkreisen aufbauen.
Durch eine Codierschaltung wird eine Darstellungsform für Zahlen oder Zeichen in eine andere Darstellungsform umgewandelt.
Bild 3.41 zeigt eine einfache Codierung Dezimaldarstellung → BCD-Code. Eine Dezimalziffer wird durch den ihr zugeordneten Schalter dargestellt. Beim BCD-Code bildet man duale Bit-Kombinationen mit 4 Dualstellen.
Aus Bild 3.42 ist die Schaltung des Bausteins *74147* zur Umwandlung einer Dezimaldarstellung 1 aus 10 in eine BCD-Darstellung mit 4 Bit und aus Tabelle 3.10. die Funktionstabelle des Bausteins *74147* zu ersehen.

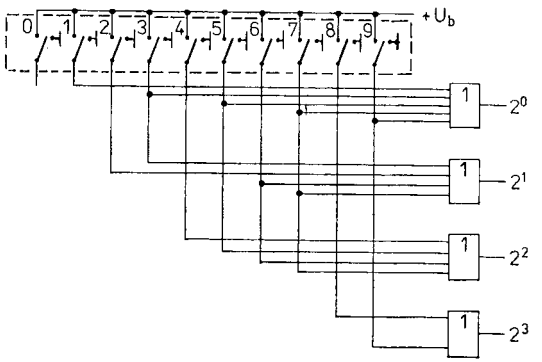


Bild 3.41 Codierung Dezimal → BCD-Code

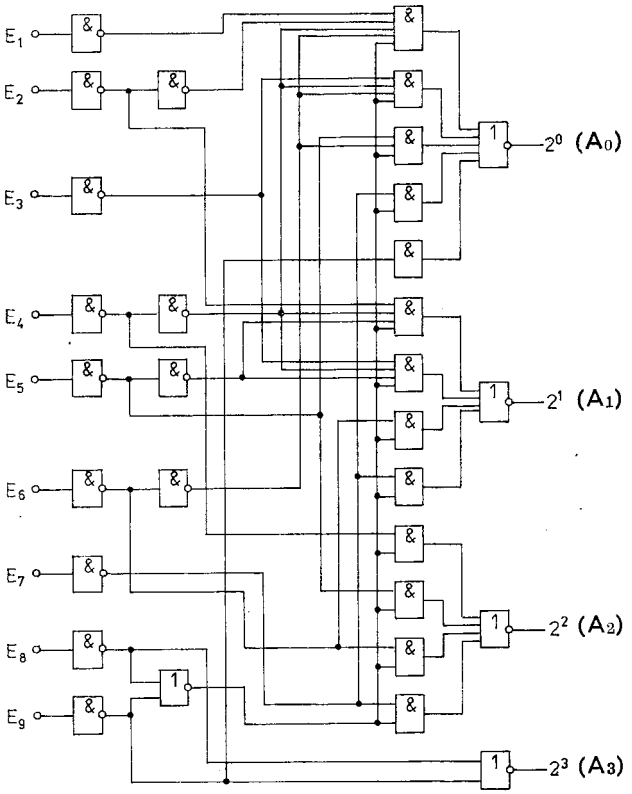


Bild 3.42 Aufbau des Bausteins *74147* zur Codierung Dezimal → BCD-Code

Der Baustein *7442* (Bild 3.43) decodiert binäre Zahlen zu Dezimalzahlen.

In der Elektronik wird sehr häufig die 7-Segment-Anzeige verwendet. Aus Bild 3.44 ist die Zifferndarstellung bei der 7-Segment-Anzeige zu ersehen. Ein 7-Segment-Decoder muß aus dem BCD-Code die Signale zur Ansteuerung der 7 Segmente liefern. Tabelle 3.11. zeigt die dazugehörige Funktionstabelle für die

Tabelle 3.10. Funktionstabelle des Bausteins 74147 (Codierer: Dezimal → BCD)

E1	E2	E3	E4	E5	E6	E7	E8	E9	A3	A2	A1	A0
H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	L	L	H	H	L
H	H	H	H	H	H	H	L	H	L	H	H	H
H	H	H	H	H	H	L	H	H	H	L	L	L
H	H	H	H	H	L	H	H	H	H	L	L	H
H	H	H	H	L	H	H	H	H	H	L	H	L
H	H	H	L	H	H	H	H	H	H	L	H	H
H	H	L	H	H	H	H	H	H	H	H	L	L
H	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

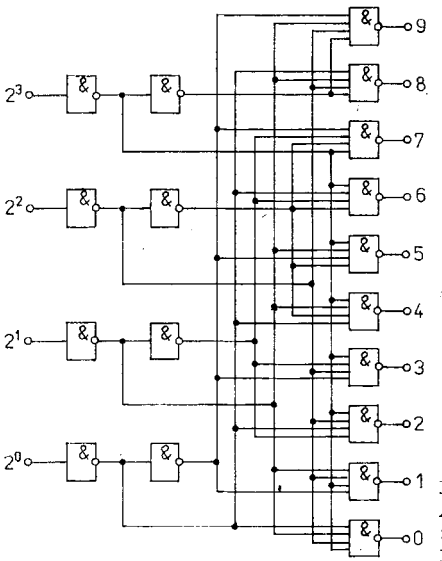


Bild 3.43
Aufbau des Bausteins *7442* zur Umwandlung BCD → Dezimal

Bild 3.44 Zifferndarstellung der Ziffern 0 bis 9 einer 7-Segment-Anzeige

Ziffern 0 bis 9. Aus dieser Funktionstabelle ergibt sich über die Kanonische Alternative Normalform

$a = P \vee D \vee AC \vee \overline{A}\overline{C}$;

$b = \overline{C} \vee D \vee AB \vee \overline{A}\overline{B}$;

$c = A \vee \overline{B} \vee C \vee D$;

$d = D \vee B\overline{C} \vee \overline{A}B \vee \overline{A}\overline{C} \vee A\overline{B}C$;

$e = \overline{A}\overline{C} \vee \overline{A}B$;

$f = D \vee \overline{A}C \vee \overline{B}C \vee \overline{A}\overline{B}$;

$g = D \vee B\overline{C} \vee \overline{A}B \vee \overline{B}C$.

Bild 3.45 zeigt die Schaltung eines 7-Segment-Decoders für die Ziffern 0 bis 9.

Tabelle 3.11. Funktionstabelle eines 7-Segment-Decoders für die Dezimalziffern 0 bis 9

Ziffer	Eingänge				Segmente						
	D	C	B	A	a	b	c	d	e	f	g
0	L	L	L	L	H	H	H	H	H	H	H
1	L	L	L	H	L	H	H	L	L	L	L
2	L	L	H	L	H	H	L	H	H	L	H
3	L	L	H	H	H	H	H	H	L	L	H
4	L	H	L	L	L	H	H	L	L	H	H
5	L	H	L	H	H	L	H	H	L	H	H
6	L	H	H	L	H	L	H	H	H	H	H
7	L	H	H	H	H	H	H	L	L	L	L
8	H	L	L	L	H	H	H	H	H	H	H
9	H	L	L	H	H	H	H	H	L	H	H

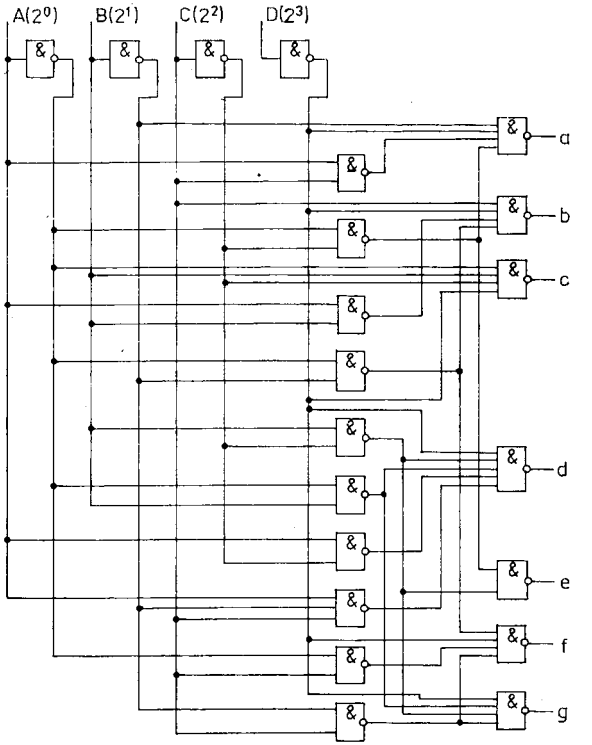


Bild 3.45 7-Segment-Decoder für die Ziffern 0 bis 9

In ähnlicher Weise läßt sich auch ein 7-Segment-Decoder für die Hexadezimalziffern 0 1 2 3 4 5 6 7 8 9 A B C D E F entwickeln. Bild 3.46 zeigt die Zeichendarstellung 0 bis F, Bild 3.47 den Aufbau des 7-Segment-Decoders, und aus Tabelle 3.12. ist die dazugehörige Funktionstabelle zu ersehen.

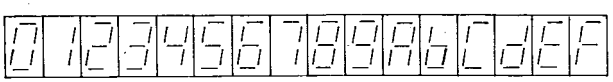


Bild 3.46 Darstellung der Hexadezimalziffern 0 bis F

Tabelle 3.12. Funktionstabelle eines 7-Segment-Decoders für die Hexadezimalziffern 0 bis F

Ziffer	Eingänge				Segmente						
	D	C	B	A	a	b	c	d	e	f	g
0	L	L	L	L	H	H	H	H	H	H	L
1	L	L	L	H	L	H	H	L	L	L	L
2	L	L	H	L	H	H	L	H	H	L	H
3	L	L	H	H	H	H	H	H	L	L	H
4	L	H	L	L	L	H	H	L	L	H	H
5	L	H	L	H	H	L	H	H	L	H	H
6	L	H	H	L	H	L	H	H	H	H	H
7	L	H	H	H	H	H	H	L	L	L	L
8	H	L	L	L	H	H	H	H	H	H	H
9	H	L	L	H	H	H	H	H	L	H	H
A	H	L	H	L	H	H	H	L	H	H	H
B	H	L	H	H	L	L	H	H	H	H	H
C	H	H	L	L	H	L	L	H	H	H	L
D	H	H	L	H	L	H	H	H	H	L	H
E	H	H	H	L	H	L	L	H	H	H	H
F	H	H	H	H	H	L	L	L	H	H	H

Für die Ansteuerung der Segmente gilt:

$a = \overline{A}\overline{C} \vee BC \vee \overline{A}B \vee B\overline{D} \vee \overline{B}C\overline{D} \vee A\overline{C}\overline{D}$;

$b = \overline{A}\overline{C} \vee \overline{B}\overline{C} \vee AB\overline{D} \vee A\overline{B}D \vee \overline{A}\overline{B}\overline{C}$;

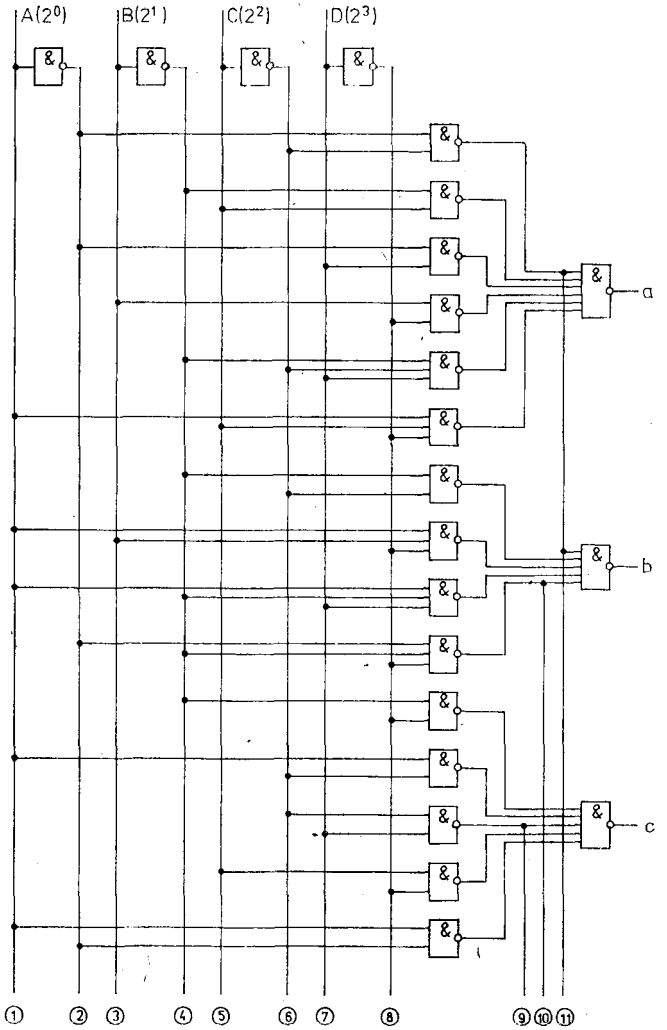
$c = \overline{B}\overline{D} \vee A\overline{C} \vee \overline{C}D \vee C\overline{D} \vee A\overline{B}$;

$d = \overline{B}D \vee AB\overline{C} \vee A\overline{B}C \vee \overline{A}BC \vee \overline{A}\overline{C}\overline{D}$;

$e = BD \vee \overline{A}B \vee CD \vee \overline{A}\overline{C}$;

$f = \overline{A}\overline{C} \vee BD \vee \overline{C}D \vee \overline{B}C\overline{D} \vee \overline{A}B\overline{D}$;

$g = B\overline{C} \vee \overline{C}D \vee AD \vee \overline{A}B \vee \overline{B}C\overline{D}$.


A(2⁰)
B(2¹)
C(2²)
D(2³)
&
a
b
c
①
②
③
④
⑤
⑥
⑦
⑧
⑨
⑩
⑪

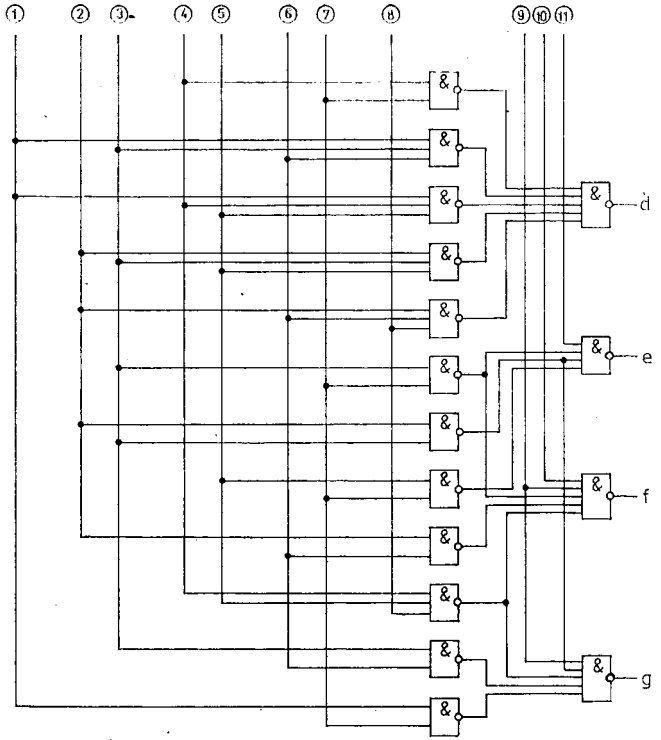


Bild 3.47 7-Segment-Decoder für die Hexadezimalziffern 0 bis F

Die Bausteine *7446*, *7447*, *7446 A* und *7447 A* sind integrierte 7-Segment-Decoder.
Bild 3.48 zeigt die Hexadezimaldarstellung der Bausteine *7446*, *7447*, *7446 A*, *7447 A*, Bild 3.49 den Aufbau der Decoder.
RBI = L ergibt eine automatische Unterdrückung der Nullan-

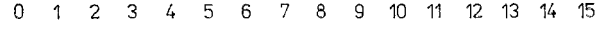


Bild 3.48 Darstellung der Hexadezimalziffern für die Bausteine *7446* und *7447*

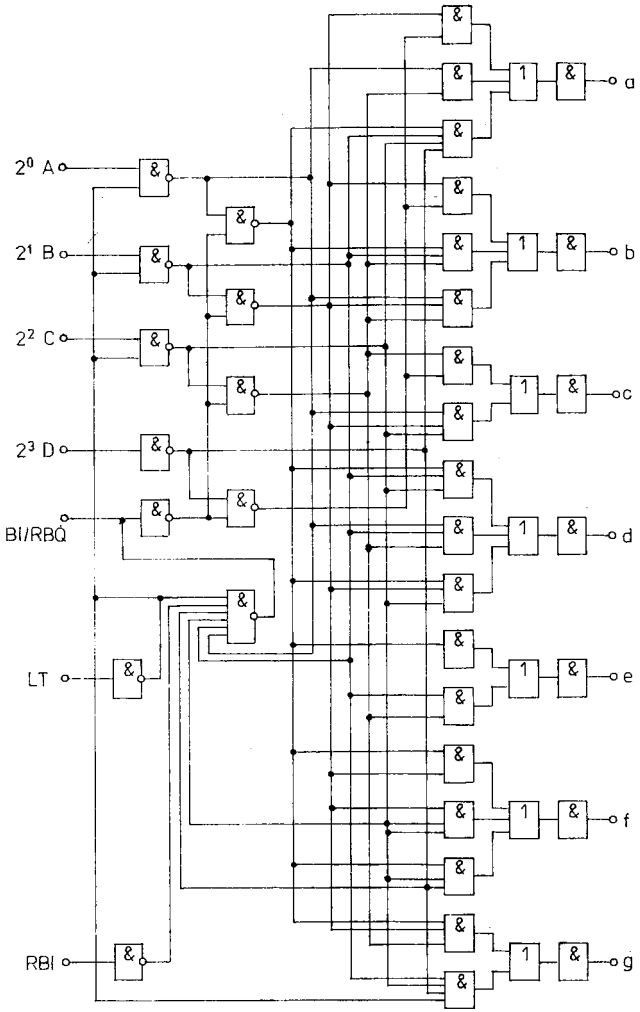


Bild 3.49 Schaltbild der 7-Segment-Decoderbausteine *7446* und *7447*

zeige. Die Segmentausgänge erhalten H-Signal, wenn die Eingänge A, B, C und D L-Signal haben (Nullbedingung).
Bei LT = L (Lampentest) führen alle Segmentausgänge L-Signal (Helltastung).
Alle 4 Bausteine unterscheiden sich bei gleicher Innenschaltung lediglich in ihren Endstufen.

7446:	offener Kollektorausgang	30 V/20 mA;
7447:	offener Kollektorausgang	15 V/20 mA;
7446 A:	offener Kollektorausgang	30 V/40 mA;
7447 A:	offener Kollektorausgang	15 V/40 mA.

Die Spannung der heute verwendeten GaAs-Leuchtdioden beträgt etwa $U_D = 1{,}6$ V. Da der vorgegebene Strom I nicht überschritten werden darf, sind zwischen Segmentanzeige und Decoder Widerstände zu schalten (Bild 3.50). Die Widerstände R können nach der Gleichung

$$R = \frac{U_b - U_D}{I};$$

$U_b = 5$ V; $U_D = 1{,}6$ V; I = angegebener höchster Wert des Stromes;

berechnet werden.
Oft liefern bestimmte Meßgeräte Ziffern in einer anderen Darstellung, als sie benötigt werden. Dann sind sogenannte „Umcodierer" notwendig. Es gibt folgende Umcodierer:
BCD-Code → Aiken-Code;
BCD-Code → 3-Exzeß-Code;
Aiken-Code → BCD-Code;
3-Exzeß-Code → BCD-Code .

Prüfbiterzeugung — Paritätsprüfer

Die häufigste Methode zur Feststellung von Fehlern bei einer Datenübertragung ist die Prüfbitkontrolle. Dabei wird die Anzahl der zur Übertragung verwendeten Bit durch ein Zusatzbit,

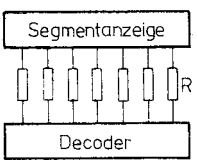


Bild 3.50
Zusammenschaltung von 7-Segment-Decoder und -Anzeige

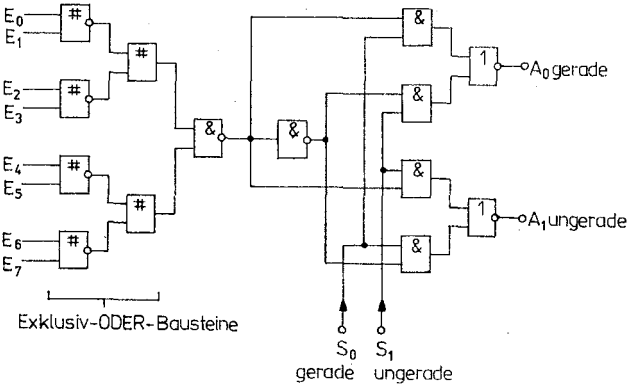


Bild 3.51 Aufbau des Bausteins *74180* zur Prüfbitbildung

das sogenannte *Prüfbit*, ergänzt. Dieses Prüfbit wird nun so gesetzt, daß die Anzahl der Einsen im Gesamtwort (Information + Prüfbit) gerade oder ungerade ist. Im Sender setzt man das Prüfbit dazu, im Empfänger läßt sich dann die gerade oder ungerade Parität überprüfen.

Bild 3.51 zeigt den Aufbau des Bausteins *74180*, der zur Überprüfung und Erzeugung der Prüfbits eines 8-Bit-Wortes (7 Bit + Prüfbit) verwendet werden kann. Aus Tabelle 3.13. läßt sich die zugehörige Funktionstabelle ersehen.

Soll z. B. zu einem 7-Bit-Wort ein 8. Bit so gesetzt werden, daß geradzahlige Parität entsteht, so verbindet man die 7 Bit mit E_0

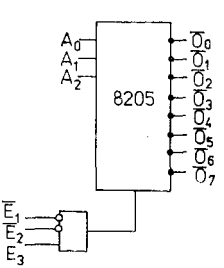
Tabelle 3.13. Funktionstabelle des Paritätsbausteins 74180

$E_0 - E_7$	S_0	S_1	A_0	A_1
gerade	H	L	H	L
ungerade	H	L	L	H
gerade	H	L	L	H
ungerade	L	H	H	L
beliebig	H	H	L	L
beliebig	L	L	H	H

Tabelle 3.14. Zusammenstellung einiger Codierschaltkreise

Typ		Kurzcharakteristik
MH 74154	ČSSR	4-Bit-Binär-Decoder (4 zu 16)
K 155 ID 4	UdSSR	2× 2-Bit-Binär-Decoder (2 zu 4)
K 155 ID 7	UdSSR	3-Bit-Binär-Decoder (3 zu 8); *Schottky*-TTL
K 500 ID 161	UdSSR	3-Bit-Binär-Decoder (3 zu 8); ECL, invertierender Ausgang
K 500 ID 162	UdSSR	3-Bit-Binär-Decoder (3 zu 8); ECL
K 155 PR 6	UdSSR	BCD/Binär-Konverter; offener Kollektorausgang
K 155 PR 7	UdSSR	Binär/BCD-Konverter; offener Kollektorausgang
7443 APC	UVR	3-Exzeß/Dezimal-Decoder
MH 7746	ČSSR	BCD/7-Segment-Decoder; Treiber, offener Kollektorausgang (30 V, 20 mA)
MH 7747	ČSSR	BCD/7-Segment-Decoder; Treiber, offener Kollektorausgang (15 V, 20 mA)
MH 7442	ČSSR	BCD/Dezimal-Decoder
7449 PC	UVR	BCD/7-Segment-Decoder; Treiber, offener Kollektorausgang 5,5 V
K 155 IP 2	UdSSR	9-Bit-Paritätsgenerator; 8-Bit-Paritätsprüfer
MH 74150	ČSSR	16-zu-1-Multiplexer; invertierender Ausgang
MH 74151	ČSSR	8-zu-1-Multiplexer
K 155 KP 5	UdSSR	8-zu-1-Multiplexer; invertierender Ausgang
K 155 KP 2	UdSSR	2×4-zu-1-Multiplexer
K 500 ID 164	UdSSR	8-zu-1-Multiplexer; ECL

bis E_6, setzt $E_7 = 0$, $S_0 = 1$ und $S_1 = 0$. Damit wird $A_1 = 1$, wenn E_0 bis E_6 ungerade sind, und $A_1 = 0$, wenn E_0 bis E_6 gerade sind. Es gilt $A_0 = \overline{A_1}$; A_1 ist unmittelbar das gesuchte Prüfbit. Zur Prüfbitkontrolle verbindet man die 8 Datenleitungen mit E_0 bis E_7, setzt $S_0 = 1$ und $S_1 = 0$. Damit ist bei geradzahliger



$A_2A_1A_0$	$\overline{O}_0\overline{O}_1\overline{O}_2\overline{O}_3\overline{O}_4\overline{O}_5\overline{O}_6\overline{O}_7$
L L L	L H H H H H H H
L L H	H L H H H H H H
L H L	H H L H H H H H
L H H	H H H L H H H H
H L L	H H H H L H H H
H L H	H H H H H L H H
H H L	H H H H H H L H
H H H	H H H H H H H L

3.52

Bild 3.52 3-zu-8-Decoder *8205* mit Funktionstabelle

Anzahl von Einsen $A_0 = 1$, $A_1 = 0$ und bei ungeradzahliger Anzahl $A_0 = 0$, $A_1 = 1$.
Mit 2 Bausteinen *74180* läßt sich auch eine 16-Bit-Prüfbitlogik aufbauen.
Wichtige Funktionen bei der Adreßentschlüsselung sind die Umcodierungen
Binär → Oktal;
Binär → Hexadezimal.
Im Fall „Binär → Oktal" werden aus einer Oktalziffer 8 Einzelsignale (3 zu 8), im Fall „Binär → Hexadezimal" 16 Einzelsignale (4 zu 16) gebildet. Der Baustein *8205* (Bild 3.52) realisiert die Funktion 3 zu 8.
In Tabelle 3.14. sind einige Codierschaltkreise zusammengestellt.

3.6. Rechenschaltkreise

Der wichtigste Rechenschaltkreis ist der Mikroprozessor. Seine Teilfunktionen werden durch eine Reihe von Rechenschaltungen realisiert, die zusammen das Rechenwerk eines Rechners darstellen. Viele dieser Teilschaltungen sind auch getrennt als Schaltkreise in MSI-Technik ausgeführt. Sie können zum Aufbau kleiner Rechenschaltungen dienen.
Die Grundfunktion des Rechners ist die Addition. Bei der Addition im Dualsystem müssen die Summe und der Übertrag von 2 Summanden und einem eventuellen Übertrag aus dem niedrigeren

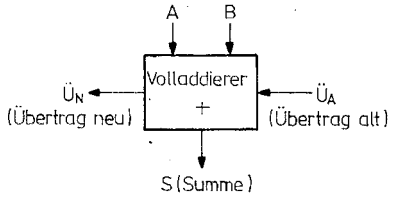


Bild 3.53
Funktion eines Volladdierers

Tabelle 3.15. Funktionstabelle eines Volladdierers

A	B	$\ddot{U}_A$	S	$\ddot{U}_N$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

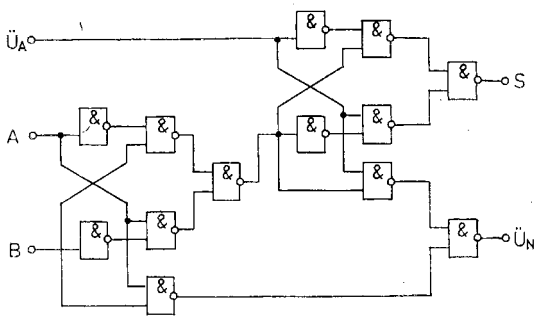


Bild 3.54 Volladdierer nach der **Funktionstabelle** Tabelle 3.15.

Stellenwert gebildet werden (Bild 3.53). Tabelle 3.15. zeigt die Funktionstabelle eines solchen Volladdierers.

Über die Kanonische Alternative Normalform ergibt sich aus Tabelle 3.15. für S und $\ddot{U}_N$ die Schaltfunktion

$S = \ddot{U}_A(\overline{A\overline{B}} \vee AB) \vee \overline{\ddot{U}}_A(\overline{A}B \vee A\overline{B})$;

$\ddot{U}_N = \ddot{U}_A(\overline{A}\overline{B} \vee A\overline{B}) \vee AB$.

Aus Bild 3.54 ist der Aufbau des dazugehörigen Volladdierers zu ersehen.

Bild 3.55 zeigt die Innenschaltung des Volladdiererbausteins *7480*. Es ist ein 1-Bit-Volladdierer mit komplementären Ein- und Ausgängen. Für die Eingänge a und b gilt:

$a^* = \overline{a_1 a_2}$ bzw. $b^* = \overline{b_1 b_2}$.

a* und b* können am Baustein abgenommen werden. Wenn a* und b* jedoch als Addiatoreingang benutzt werden, dann müssen a_1, a_2, b_1 und b_2 an Masse liegen. Für die Eingangssignale a und b gilt:

$a = \overline{a^* a_c}$; $b = \overline{b^* b_c}$.

Wenn a_1, a_2, b_1 und b_2 an Masse liegen, so sind $a^* = b^* = 1$ und $a = \overline{a_c}$, $b = \overline{b_c}$.

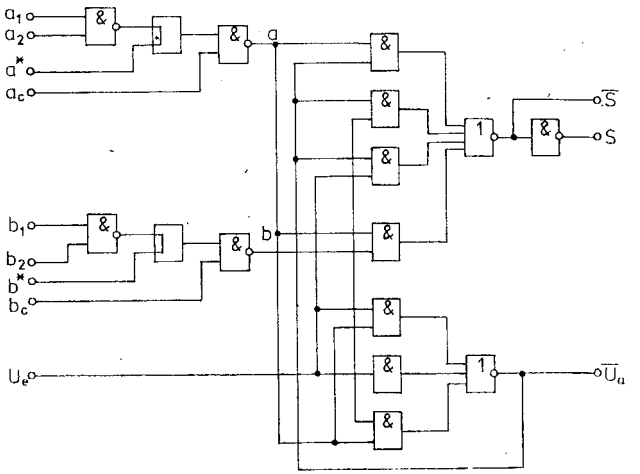


Bild 3.55 Volladdiererbaustein *7480*

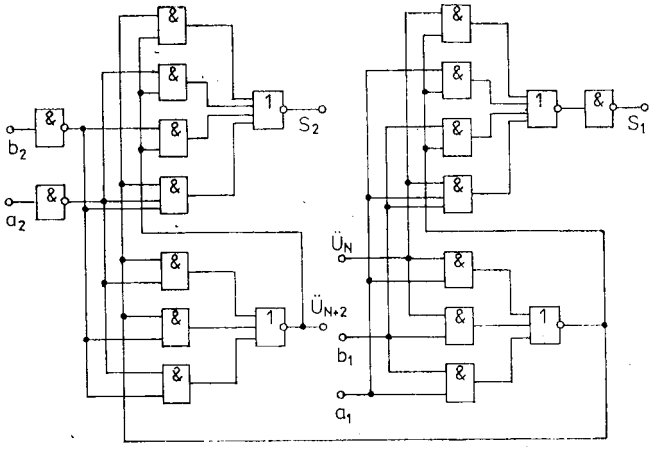


Bild 3.56 Volladdiererbaustein *7482*

Tabelle 3.16. Funktionstabelle des Volladdiererbausteins 7482

				$C_n = L$			$C_n = H$		
b_2	a_2	b_1	a_1	S_1	S_2	C_{n+2}	S_1	S_2	C_{n+2}
L	L	L	L	L	L	L	H	L	L
L	L	L	H	H	L	L	L	H	L
L	L	H	L	H	L	L	L	H	L
L	L	H	H	L	H	L	H	H	L
L	H	L	L	L	H	L	H	H	L
L	H	L	H	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
L	H	H	H	L	L	H	H	L	H
H	L	L	L	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
H	L	H	L	H	H	L	L	L	H
H	L	H	H	L	L	H	H	L	H
H	H	L	L	L	L	H	H	L	H
H	H	L	H	H	L	H	L	H	H
H	H	H	L	H	L	L	L	H	H
H	H	H	H	L	H	H	H	H	H

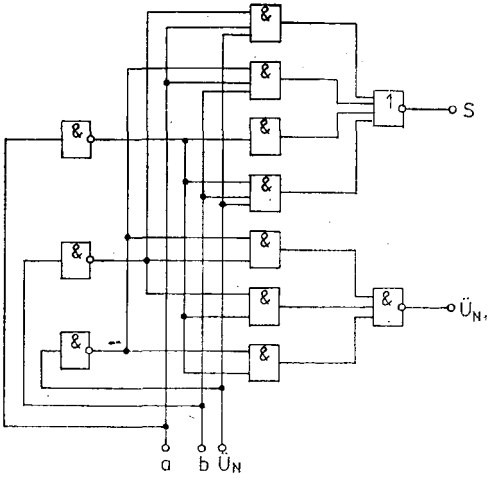


Bild 3.57 Schneller Volladdierer *74183*

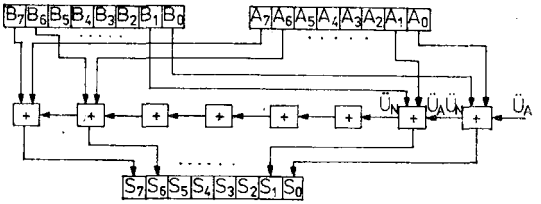


Bild 3.58 Paralleles Addierwerk

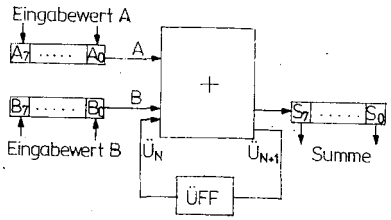


Bild 3.59
Serielles Addierwerk

Addierbausteine gibt es in sehr unterschiedlichen Ausführungen.
Beispiele
7482: 2 Volladdierer (Bild 3.56, Funktionstabelle Tabelle 3.16);
74183: schneller Volladdierer (Bild 3.57, Funktionstabelle Tabelle 3.15.).
2 Dualzahlen können mit Hilfe eines Volladdierers parallel oder seriell addiert werden. Aus Bild 3.58 ist der Aufbau des parallelen und aus Bild 3.59 der des seriellen Addierwerks zu ersehen. Die Register A, B und S (Bild 3.59) sind Schieberegister. Die Eingaberegister A und B müssen für die Paralleleingabe geeignet sein. Das Summenregister S nimmt die Daten seriell auf und gibt sie parallel ab. Mit jedem Takt werden die Register um eine Stelle nach rechts verschoben. Dabei werden die Stellen A_0, B_0 und ÜFF (Übertrags-Flip-Flop) über die Addierschaltung geführt. Die Werte der Ausgänge S und $Ü_{N+1}$ der Addierschaltung werden in S_7 und ÜFF eingetragen.
Eine sehr häufige Funktion bei Mikroprozessoren ist die Erhöhung (*Inkrementierung*) oder Subtraktion (*Dekrementierung*) einer 1 vom Inhalt eines Registers. Tabelle 3.17. enthält die Funktionstabelle

Tabelle 3.17. Funktionstabelle eines Dekrementierers

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

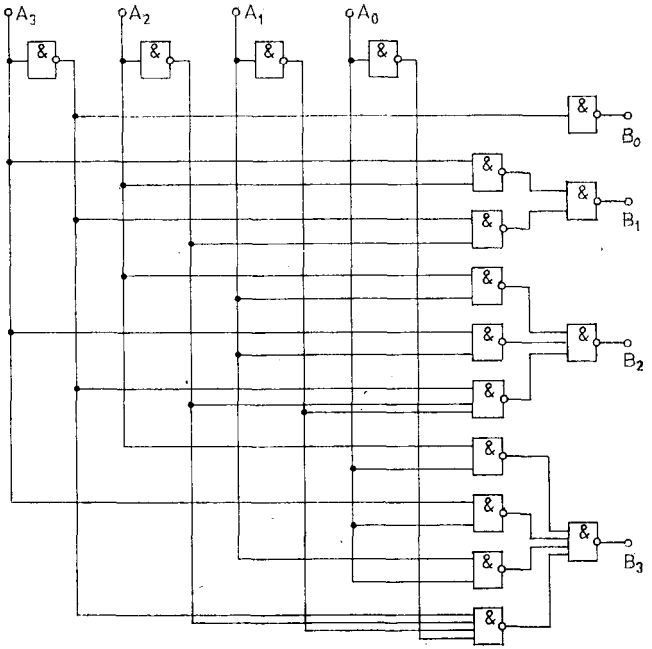


Bild 3.60 Dekrementierschaltung

Tabelle 3.18. Zusammenstellung einiger Rechenschaltkreise

Typ		Kurzcharakteristik
K 155 IM 1	UdSSR	1-Bit-Volladdierer
K 155 IM 2	UdSSR	2-Bit-Volladdierer
K 155 IM 3	UdSSR	4-Bit-Volladdierer
K 155 IP 3	UdSSR	4-Bit-Recheneinheit
7485 PC	UVR	4-Bit-Vergleicher
K 531 IP 3	UdSSR	4-Bit-Recheneinheit; *Schottky*-TTL
K 500 IP 181	UdSSR	4-Bit-Recheneinheit; ECL

einer Dekrementierung. Über die Kanonische Alternative Normalform ergibt sich aus Tabelle 3.17.:

$B^0 = \overline{A}_0$;

$B_1 = A_0A_1 \vee \overline{A}_0\overline{A}_1$;

$B_2 = A_1A_2 \vee A_0A_2 \vee \overline{A}_0\overline{A}_1\overline{A}_2$;

$B_3 = A_1A_3 \vee A_0A_3 \vee A_2A_3 \vee \overline{A}_0\overline{A}_1\overline{A}_2\overline{A}_3$.

Aus Bild 3.60 ist der Aufbau eines Dekrementierers zu ersehen. In Tabelle 3.18. sind einige Rechenschalkreise zusammengestellt.

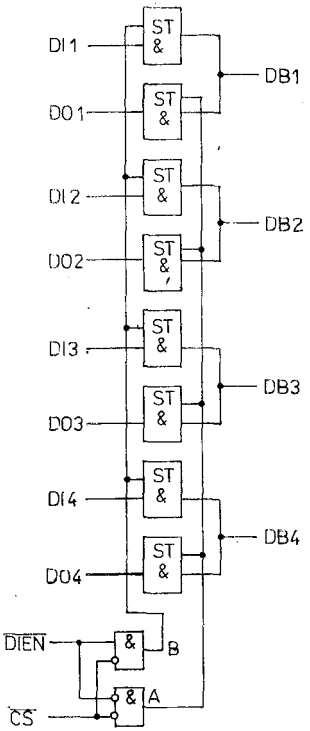


Bild 3.61
Bustreiberbaustein *8216*

Tabelle 3.19. Funktionstabelle des Bustreiberbausteins 8216

$\overline{CS}$	$\overline{DIEN}$	
L	L	DI → DB
L	H	DB → DO
H	L	hochohmig
H	H	hochohmig

3.7. Bustreiber

Die aus den Schaltkreisen kommenden Signale haben nicht immer die Leistung, die notwendig ist, um mehrere angeschlossene Bausteine zu steuern. Teilweise sind auch die Mindestpotentiale zur Steuerung eines Bausteins größer als die Potentiale, die der Geberschaltkreis liefert. Zur Anpassung der Potentiale und der Leistungsparameter im Mikrorechnerbus dienen sogenannte Bustreiber. Da der Mikrorechnerbus meistens in beiden Richtungen betrieben wird, arbeiten diese Bustreiber „bidirektional".
Bild 3.61 zeigt den Aufbau des Bustreiberbausteins *8216*. Wenn bei A H-Potential vorliegt, so arbeitet der Baustein in Richtung DI → DB und bei H-Potential an B in Richtung DB → DO. Liegt A bzw. B auf L-Potential, so sind die Ausgänge der entsprechenden Treiber hochohmig.
Die Richtung des Informationsflusses wird durch das Signal $\overline{DIEN}$ bestimmt, während man mit $\overline{CS}$ (chip-select) den Baustein freigibt. Aus Tabelle 3.19. ist die Funktionstabelle des Bausteins *8216* zu ersehen.
Bild 3.62 zeigt die Verwendung von 2 Bausteinen *8216* zur Steuerung eines 8-Bit-Datenbusses. Während die Eingangsspannungen des Bausteins *8216* für H-Potential nur mindestens 2 V betragen müssen, liefert der Baustein ein H-Potential von mindestens 3,65 V.

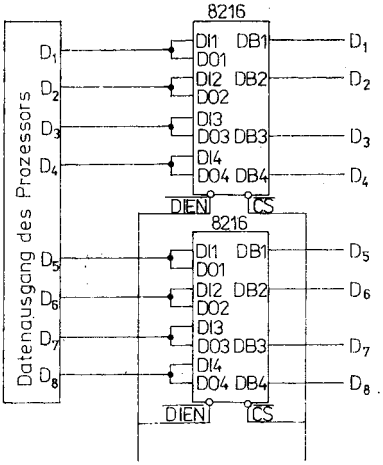


Bild 3.62
Steuerung eines 8-Bit-Datenbusses mit 2 × *8216*

3.8. Zähler

Zähler lassen sich auf verschiedenste Art aufbauen. Verwendet man getriggerte Flip-Flop, die mit der HL-Flanke umkippen, und verbindet den jeweiligen Ausgang der vorherigen Zählstufe mit dem Takteingang des nächsten Flip-Flop (Bild 3.63), so spricht man von *asynchronen Zählern.* Wird der Takt an alle Flip-Flop gleichzeitig angelegt und eine logische Verbindung zwischen den Ausgängen der vorangehenden Flip-Flop und den Eingängen des nächsten Flip-Flop geschaffen, so erhält man *synchrone Zähler.* Ein Beispiel eines synchronen Vorwärtszählers zeigt Bild 3.64.

Zähler mit JK-Flip-Flop

Jedes als Trigger geschaltete JK-Flip-Flop teilt die Eingangsfrequenz im Verhältnis 1:2. Bild 3.65 zeigt die Schaltung für einen Vorwärtszähler im Dualcode und das zugehörige Impulsdia-

Bild 3.63
Schaltprinzip eines asynchronen Zählers

Tabelle 3.20. Zusammenstellung einiger Treiberschaltkreise

Typ		Kurzcharakteristik
K 155 LN 3	UdSSR	6 Treiber, invertiert, offener Kollektorausgang, 30 V
K 155 LN 4	UdSSR	6 Treiber, offener Kollektorausgang, 30 V
7416 PC	UVR	6 Treiber, invertiert, offener Kollektorausgang, 15 V
7417 PC	UVR	6 Treiber, offener Kollektorausgang, 15 V
74125 PC	UVR	4 Bustreiber, Tri-state
74126 PC	UVR	4 Bustreiber, Tri-state
SN 74367 AN		6 Bustreiber, 2 Enable-Eingänge, Tri-state

gramm. Soll der Zähler bereits nach Stellung 9 zurückschalten, so kann eine Schaltung nach Bild 3.66 verwendet werden. Durch eine entsprechende Zählstruktur, d. h. durch Schalten der einzelnen Setz- oder Rücksetzbedingungen, lassen sich beliebige Zählfunktionen realisieren. Zum Beispiel benötigt man für einen Zähler im 3-Exzeß-Code die in Tabelle 3.21. stehende Zählfolge. Bild 3.67 zeigt das dazugehörige Schaltbild mit dem Taktdiagramm.

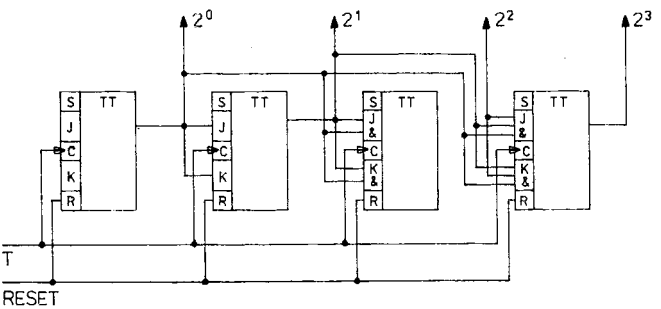


Bild 3.64 Synchroner Zähler

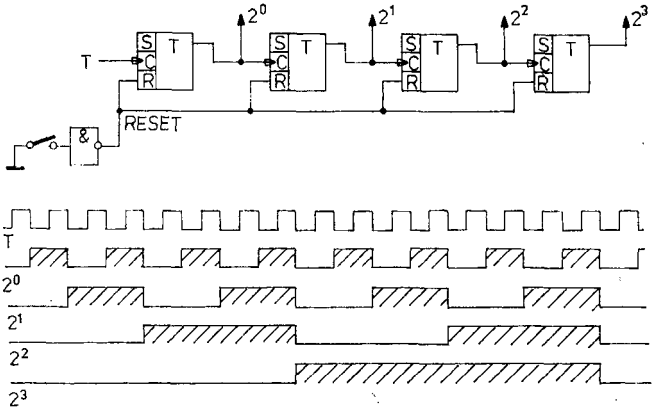


Bild 3.65 Vorwärtszähler im Dualcode mit dem dazugehörigen Impulsdiagramm

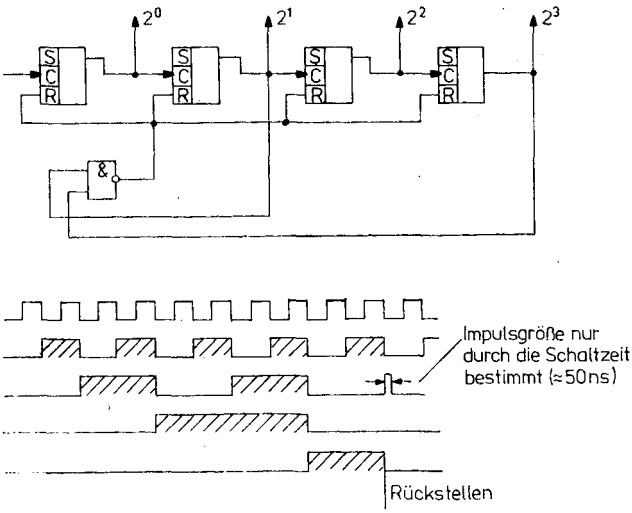


Bild 3.66 Vorwärtszähler im BCD-Code

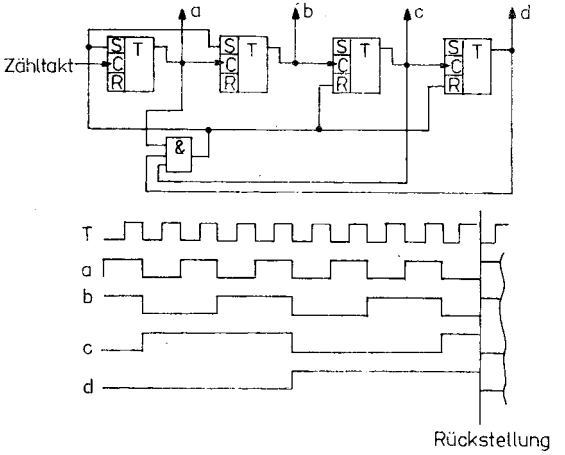


Bild 3.67 Zähler im 3-Exzeß-Code (Das gezeichnete AND-Gatter ist ein NAND!)

Tabelle 3.21. Zählfolge für einen Zähler im 3-Exzeß-Code

	d	c	b	a
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

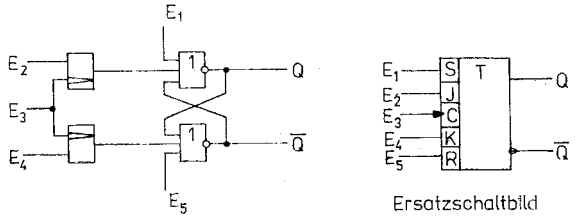


Bild 3.68 RS-Flip-Flop mit dynamischer Ansteuerung

Zähler mit RS-Flip-Flop

Schaltet man vor die Eingangsstufen eines RS-Flip-Flop Dynamikvorsätze (Kondensatorspeicher), so lassen sich sehr einfach Zählschaltungen aufbauen.

Der in Bild 3.68 dargestellte Speicher mit dynamischer Ansteuerung eignet sich als Baustein zum Aufbau von Zählstufen. Bild 3.69 zeigt einen vierstelligen Dualzähler. Die einzelnen Stufen sind als Teiler geschaltet. Jede Stufe wird von der Vorstufe angesteuert, wenn deren Ausgangssignal von H nach L springt. Mit TL werden die Zählstufen in die Ausgangsstellung 0000 gebracht.

3.9. Taktgeneratoren

Die meisten Rechnerschaltkreise arbeiten taktgesteuert. Bild 3.70 zeigt eine einfache Variante zur Erzeugung einer unstabilisierten Taktserie. Liegt am Punkt A H-Pegel, so hat Punkt B L-Pegel. Der Kondensator C entlädt sich über R, bis Punkt A L-Pegel erhält. Jetzt hat B H-Pegel, und der Kondensator C lädt sich über

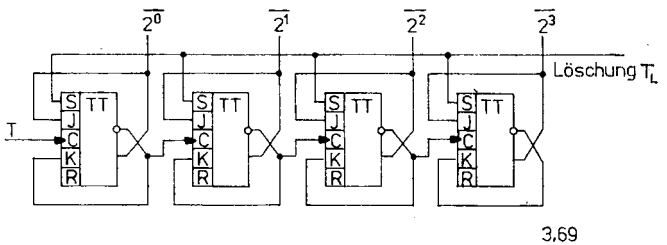


Bild 3.69 Vierstelliger Dualzähler mit Schaltelementen nach Bild 3.68

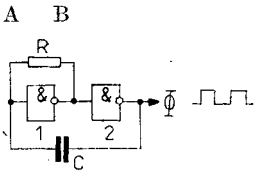


Bild 3.70
Erzeugung einer unstabilisierten Taktserie

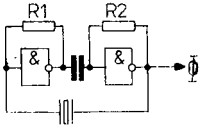


Bild 3.71
Einfacher Taktgenerator

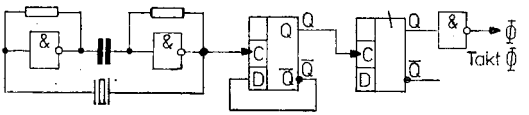


Bild 3.72 Taktgenerator mit zweifacher Untersetzung

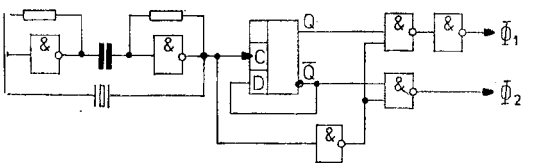


Bild 3.73 Taktgenerator zur Erzeugung von 2 Taktserien Φ_1 und Φ_2

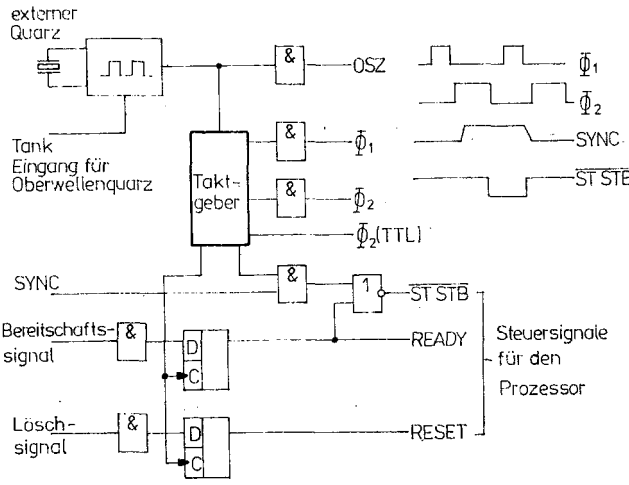


Bild 3.74 Integrierter Taktgenerator *8224* für den Prozessor *8080*

R und den Negator 1 wieder auf. Die Umladung von C wird durch Negator 2 geringfügig unterstützt.

Bild 3.71 zeigt eine Schaltung, in der die Auf- und Entladung von C durch R_1 und R_2 geschieht. Außerdem wird durch Hinzuschalten eines Quarzes die erzeugte Taktserie frequenzstabilisiert. Liegt die Quarzfrequenz zu hoch, so kann man die Taktfrequenz mit D-Flip-Flop untersetzen. Bild 3.72 zeigt eine solche Untersetzung mit 2 D-Flip-Flop.

Sollen aus dem Urtakt 2 Taktserien hergestellt werden, so läßt sich das mit einem D-Flip-Flop nach Bild 3.73 erreichen.

Der Baustein *8224* ist ein integrierter Taktgenerator für den Mikroprozessor *8080*. Er erzeugt die notwendigen Taktimpulse Φ_1 und Φ_2 und dient gleichzeitig zur Verarbeitung des Zeitsignals SYNC und zur Erzeugung der Signale RESET, READY und $\overline{\text{STSTB}}$ (Bild 3.74.).

In Tabelle 3.22. sind einige Zählschaltkreise zusammengestellt.

Tabelle 3.22. Zusammenstellung einiger Zählerschaltkreise

Typ		Kurzcharakteristik
MH 7490	ČSSR	Dezimalzähler
K 155 IE 9	UdSSR	programmierbarer Dezimalzähler
74176 PC	UVR	programmierbarer Dezimalzähler
74190 PC	UVR	Synchroner programmierbarer Vorwärts-/Rückwärts-Zähler (Dezimalzähler)
D 192 C/D	DDR	Synchroner programmierbarer Vorwärts-/Rückwärts-Zähler (Dezimalzähler)
74290 PC	UVR	Dezimalzähler
MH 7493	ČSSR	4-Bit-Binärzähler
74161 PC	UVR	programmierbarer 4-Bit-Binärzähler
74177 PC	UVR	programmierbarer 4-Bit-Binärzähler
74191 PC	UVR	Synchroner programmierbarer Vorwärts-/Rückwärts-Zähler (4-Bit-Binärzähler)
D 193 C/D	DDR	Synchroner programmierbarer Vorwärts-/Rückwärts-Zähler (4-Bit-Binärzähler)
74293 PC	UVR	4-Bit-Dualzähler

Anhang

1: Darstellung der Zahlen 0 bis 32 in dezimaler, dualer, oktaler, hexadezimaler und BCD-Form

dezimal	dual	oktal	hexa-dezimal	BCD
0	000000	0	0	0000 0000
1	000001	1	1	0000 0001
2	000010	2	2	0000 0010
3	000011	3	3	0000 0011
4	000100	4	4	0000 0100
5	000101	5	5	0000 0101
6	000110	6	6	0000 0110
7	000111	7	7	0000 0111
8	001000	10	8	0000 1000
9	001001	11	9	0000 1001
10	001010	12	A	0001 0000
11	001011	13	B	0001 0001
12	001100	14	C	0001 0010
13	001101	15	D	0001 0011
14	001110	16	E	0001 0100
15	001111	17	F	0001 0101
16	010000	20	10	0001 0110
17	010001	21	11	0001 0111
18	010010	22	12	0001 1000
19	010011	23	13	0001 1001
20	010100	24	14	0010 0000

dezimal	dual	oktal	hexa-dezimal	BCD
21	010101	25	15	0010 0001
22	010110	26	16	0010 0010
23	010111	27	17	0010 0011
24	011000	30	18	0010 0100
25	011001	31	19	0010 0101
26	011010	32	1A	0010 0110
27	011011	33	1B	0010 0111
28	011100	34	1C	0010 1000
29	011101	35	1D	0010 1001
30	011110	36	1E	0011 0000
31	011111	37	1F	0011 0001
32	100000	40	20	0011 0010

2: Rechnen mit Dualzahlen

Addition

Während bei Dezimalzahlen ein Übertrag entsteht, wenn die Summe zweier Ziffern größer als 9 ist, tritt im Dualsystem der Übertrag bereits auf, wenn die Summe größer als 2 ist. Im Dualsystem gelten folgende Grundregeln:

$0 + 0 = 0$ mit Übertrag 0;
$0 + 1 = 1$ mit Übertrag 0;
$1 + 0 = 1$ mit Übertrag 0;
$1 + 1 = 0$ mit Übertrag 1.

Beispiel

Bilden der Summe $13 + 7$.

		1	Übertrag				1111	Übertrag
Dezimal:	13			Dual:	13	$\triangleq$	1101	
	+ 7				+ 7	$\triangleq$	111	
	20				20	$\triangleq$	10100	

Subtraktion

Gelingt die Subtraktion in einer Stelle nicht direkt, so muß der Minuend durch „Borgen" aus der nächsthöheren Stelle erhöht werden. Im Dualsystem wird das „Borgen" aus der nächsthöheren Stelle dadurch realisiert, daß von der höherwertigen Stelle, wie auch im Dezimalsystem, eine 1 abgezogen und dafür die betreffende Stelle um 10 erhöht wird.

Beispiel

Bilden der Differenz 91 − 53.

	1 geborgt							1 geborgt
Dezimal:	91	Dual:	91 ≙	1	0	1	1	0	1	1
	− 53		− 53 ≙		1	1	0	1	0	1
	38		38 ≙		1	0	0	1	1	0

Werden negative Dualzahlen im Komplement dargestellt, dann realisiert man die Subtraktion durch die Addition im Zweierkomplement. Die Zahl − 53 sieht in Zweierkomplementdarstellung folgendermaßen aus:

− 53 ≙ 1 1 0 0 1 0 1 1 ,

wobei das vorderste Bit die Vorzeichenstelle ist.

Das Beispiel 91–53 sieht, wenn statt 91 − 5391 + (− 53) gerechnet wird, so aus:

91 ≙	0	1	0	1	1	0	1	1
− 53 ≙	1	1	0	0	1	0	1	1
	0	0	1	0	0	1	1	0

Der Übertrag in die Stelle vor dem Vorzeichen wird dabei nicht mehr berücksichtigt, da er über die vorgegebene Stellenzahl hinausgeht. Zur Begründung der angewandten Methode setzt man für eine negative Zahl y in der Komplementdarstellung das Zeichen y'. Dann ist $y' = y + k_2$, mit $k_2 = 2^8$, wenn es sich um das Zweierkomplement handelt.

Die Zahl $k_2 = 2^8$ läßt sich mit den vorgegebenen Stellen nicht darstellen, da der höchste Stellenwert 2^7 ist.

Für $x - y$ gilt:

$x - y = x + (- y) = x + y' = x - y + k_2$.

Ist $x - y$ negativ, so steht das Ergebnis mit $x - y + k_2$ richtig im Register. Ist $x - y$ positiv, so ergibt sich ebenfalls ein richtiges Ergebnis, da die Stelle für k_2 nicht vorhanden ist.

Sind x und y negativ, dann gilt:
$-x-y=x'+y'=-x-y+2k_2=-x-y+k_2+k_2$.
Der Wert $-x-y+k_2$ ist die Komplementdarstellung von $-x-y$. Das zweite k_2 wird nicht im Register dargestellt, da die daf ür notwendige Stelle nicht vorhanden ist.

Multiplikationen

Zur Multiplikation müssen die beiden Operanden positiv sein. Die Multiplikation von positiven Zahlen gleicht der im Dezimalsystem, wobei folgende Grundregeln gelten:
$0 \cdot 0 = 0$;
$0 \cdot 1 = 0$;
$1 \cdot 0 = 0$;
$1 \cdot 1 = 1$.
Die Multiplikation ist eine fortgesetzte Addition, wobei die einzelnen Summanden entsprechend ihrem Stellenwert verschoben sind.
Beispiel
Bilden des Produkts 12 * 5.

```
1100 * 101
 0000
  1100
------
111100 ≙ 60
```

Auch hier kann wie beim Rechnen mit Dezimalzahlen im angegebenen Beispiel die Nullzeile weggelassen werden.

Division

Die Division ist im wesentlichen eine fortgesetzte Subtraktion. In den Quotienten wird dann eine Ziffer 1 eingetragen, wenn der Subtrahend kleiner als der Minuend ist. Da die Subtraktion auch als Addition im Komplement realisiert werden kann, ergeben sich unterschiedliche Verfahren für die Division, die jedoch im Prinzip auf der gleichen Grundlage beruhen.

Beispiel 1
Bilden der Division 108:12 nach dem normalen Handrechenverfahren.

```
1101100:1100 = 1001
1100         geht zu substrahieren
             (Subtrahend ist kleiner als
             Minuend), Ergebnis ist 1;
0001100      Rest
```

Divisor 1 Stelle nach rechts	− 1 1 0 0	geht nicht zu subtrahieren, Ergebnis ist 0;
	0 0 0 1 1 0 0	Rest
Divisor 1 Stelle nach rechts	− 1 1 0 0	geht nicht zu subtrahieren, Ergebnis ist 0;
	0 0 0 1 1 0 0	Rest
Divisor 1 Stelle nach rechts	− 1 1 0 0	geht zu subtrahieren, Ergebnis ist 1;
	0 0 0 0 0 0 0	Rest

Da der Rest 0 ist, lautet das Ergebnis 1 0 0 1.

Beispiel 2

Bilden der Division 108 : 12 mit Rückstellung des Restes unter Verwendung der Komplementdarstellung.

1 1 0 1 1 0 0 : 1 1 0 0 = 1 0 0 1

Stellenzahl	8 7 6 5 4 3 2 1 0	
	0 1 1 0 1 1 0 0	Divident
	+ 1 0 1 0 0 0 0 0	Zweierkomplement des Divisors
	0 0 0 0 1 1 0 0	Ergebnis ist positiv — ergibt eine 1 im Quotienten
	+ 1 1 0 1 0 0 0 0	Divisor um eine Stelle nach rechts verschoben (Zweierkomplement)
	1 1 0 1 1 1 0 0	Ergebnis ist negativ — ergibt eine 0 im Quotienten
	+ 1 1 0 0	Da das vorangegangene Ergebnis negativ ist, muß der Divisor wieder dazugezählt werden (Rückstellung des Restes)
	0 0 0 0 1 1 0 0	
	+ 1 1 1 0 1 0 0 0	Divisor um eine Stelle nach rechts verschoben,
	1 1 1 1 0 1 0 0	Ergebnis ist negativ — ergibt eine 0 im Quotienten

+ 1100	Da das vorangegangene Ergebnis negativ ist, muß der Divisor
00001100	wieder dazugezählt werden (Rückstellung des Restes)
+ 11110100	Divisor um 1 nach rechts verschoben
00000000	Ergebnis positiv — ergibt 1 im Quotienten

Der verbleibende Rest ist 0, die Division geht auf und ergibt den Quotienten 1001.

Geht man von der Tatsache aus, daß Multiplikation und Division durch fortgesetzte Additionen bzw. Subtraktionen realisiert werden, so läßt sich zeigen, daß im Dualsystem weniger Additionen und Subtraktionen notwendig sind als im Dezimalsystem.

Rechnen mit codierten Zahlen

In diesem Abschnitt wird nur auf BCD-codierte Zahlen eingegangen, da der BCD-Code der am häufigsten verwendete Code ist. Für die Rechenoperationen im BCD-Code gibt es bei den Mikroprozessoren *8080* und *U 880* spezielle Befehle, Werden 2 Dezimalziffern im BCD-Code addiert, so können folgende Fälle auftreten:

1. Die Summe ist $\leqq 9$.
2. Die Summe ist > 9 aber $\leqq 15$.
3. Die Summe ist > 15.

Im Fall 1 ist die Ergebnisziffer richtig.

Beispiel

Summe 4 + 5: 4 $\triangleq$ 0100
5 $\triangleq$ 0101
9 $\triangleq$ 1001

Im Fall 2 ist die Ergebnisziffer eine Pseudotetrade. Die richtige Ziffer erhält man durch Addition von 6.

Beispiel

Summe 7 + 6: 7 $\triangleq$ 0111
6 $\triangleq$ 0110
1101 = Pseudotetrade
+ 110 (Addition 6)
13 $\triangleq$ 10011

Im Fall 3 ergibt sich bereits der richtige Übertrag, aber die Ergebnisziffer muß noch um 6 erhöht werden.

Beispiel

Summe 8 + 9: 8 $\triangleq$ 1000
9 $\triangleq$ 1001

Übertrag in die nächste Dezimalziffer ——→ 1 0001
+ 110 (Addition 6)
17 $\triangleq$ 1 0111

Für die Addition im BCD-Code lassen sich also folgende Regeln aufstellen:
Ergibt die Summe zweier Ziffern plus Übertrag von der vorherigen Stelle keine Pseudotetrade und keinen Übertrag in die nächste Dezimalstelle, so ist die Ergebnisziffer richtig.
Ergibt die Summe zweier Ziffern eine Pseudotetrade oder einen Übertrag in die nächste Stelle, dann muß die Ergebnisziffer um 6 erhöht werden.

Beispiel

Summe 2985 + 4936 = 7921.

1←	1←	1←	Übertrag in
0 0 1 0	1 0 0 1	1 0 0 0	0 1 0 1 nächste Tetrade
0 1 0 0	1 0 0 1	0 0 1 1	0 1 1 0
0 1 1 1	0 0 1 1	1 1 0 0 $\triangleq$ PS*	1 0 1 1 $\triangleq$ PS*
	1 1 0 (+ 6)	1 1 0 (+ 6)	1 1 0 (+ 6)
0 1 1 1	1 0 0 1	0 0 1 0	0 0 0 1 $\triangleq$ 7921

*PS = Pseudotetrade

Rechnen mit Gleitkommazahlen

Addition und Subtraktion von Gleitkommazahlen

Die Addition bzw. Subtraktion der Mantissen zweier Zahlen ist nur dann möglich, wenn die Exponenten beider Zahlen gleich sind.
Für die Berechnung von
$Z_1 + Z_2 = m_1 \cdot 10^{E1} + m_2 \cdot 10^{E2}$
muß unter der Voraussetzung, daß $E_1 < E_2$ ist, m_1 um $E_2 - E_1$

Stellen nach rechts verschoben werden. Die neue Mantisse sei mit m_1' bezeichnet, dann ist
$Z_1 + Z_2 = m_1' \cdot 10^{E2} + m_2 \cdot 10^{E2} = (m_1' + m_2) \cdot 10^{E2}$.
Die neue Mantisse ist also $m_1' + m_2$, der neue Exponent E_2.

Beispiel
$Z_1 = 0{,}25 \cdot 10^2$;
$Z_2 = 0{,}35 \cdot 10^4$.
$E_2 - E_1 = 2$, d. h., die Mantisse von Z_1 muß um 2 Stellen nach rechts verschoben werden. Es gilt also:
$Z_1 + Z_2 = 0{,}25 \cdot 10^2 + 0{,}35 \cdot 10^4$
$= 0{,}0025 \cdot 10^4 + 0{,}35 \cdot 10^4 = 0{,}3525 \cdot 10^4$.
Bei der Addition im Dualsystem verfährt man analog.
Bei der Subtraktion ist in entsprechender Weise vorzugehen.

Multiplikation von Gleitkommazahlen
Die Multiplikation von Gleitkommazahlen läuft nach der Regel
$Z_1 \cdot Z_2 = m_1 \cdot 10^{E1} \cdot m_2 \cdot 10^{E2} = m_1 \cdot m_2 \cdot 10^{E1+E2}$
ab. Die neue Mantisse ist also $m_1 \cdot m_2$, der neue Exponent $E_1 + E_2$. Die neue Mantisse $m_1 \cdot m_2$ darf ebenfalls keine 0 nach dem Komma aufweisen, d. h., gegebenenfalls ist die 0 nach dem Komma durch Verschiebung von $m_1 \cdot m_2$ um eine Stelle nach links zu beseitigen, wobei der Exponent $E_1 + E_2$ um 1 zu erniedrigen ist.

Beispiel 1 (Dezimalsystem)
$Z_1 = 0{,}2 \cdot 10^2$;
$Z_2 = 0{,}3 \cdot 10^3$;
$Z_1 \cdot Z_2 = 0{,}2 \cdot 0{,}3 \cdot 10^{2+3} = 0{,}06 \cdot 10^5 = 0{,}6 \cdot 10^4$.
(Die „0" nach dem Komma bei 0,06 wurde durch Verschieben um eine Stelle nach rechts und Erniedrigung des Exponenten 5 um 1 beseitigt.)
Im Dualsystem wird in entsprechender Weise verfahren.

Beispiel 2 (Dualsystem)
$3 \cdot 4 = 12$.
$3 \triangleq 11 = 0{,}11 \cdot 2^2 = 0{,}11 \cdot 2^{10}$; $4 \triangleq 100 = 0{,}1 \cdot 2^3 = 0{,}1 \cdot 2^{11}$;
$0{,}11 \cdot 2^{10} \cdot 0{,}1 \cdot 2^{11} = 0{,}11 \cdot 0{,}1 \cdot 2^{10+11} = 0{,}011 \cdot 2^{101}$;
$= 0{,}11 \cdot 2^{(101-1)} = 0{,}11 \cdot 2^{100}$.

Division von Gleitkommazahlen
Die Division von Gleitkommazahlen läuft nach der Regel

$$\frac{Z_1}{Z_2} = \frac{m_1 \cdot 10^{E1}}{m_2 \cdot 10^{E2}} = \frac{m_1}{m_2} \cdot 10^{E1-E2}$$

ab. Die neue Mantisse ist also $m_1 : m_2$, der neue Exponent $E_1 - E_2$. Da die neue Mantisse der Vorschrift $-1 < m < +1$ genügen muß, ist gegebenenfalls die Ergebnismantisse um eine Stelle nach rechts zu verschieben und der Exponent dabei um 1 zu erhöhen.

Beispiel 1 (Dezimalsystem)

$Z_1 = 0{,}25 \cdot 10^2;$

$Z_2 = 0{,}5 \cdot 10^1;$

$$\frac{Z_1}{Z_2} = \frac{0{,}25 \cdot 10^2}{0{,}5 \cdot 10^1} = \frac{0{,}25}{0{,}5} \cdot 10^{2-1} = 0{,}5 \cdot 10^1.$$

Beispiel 2 (Dualsystem)

$6 : 2 = 3.$

$Z_1 = 6 \triangleq 0{,}11 \cdot 2^{11};$

$Z_2 = 2 \triangleq 0{,}1 \cdot 2^{10};$

$$\frac{Z_1}{Z_2} = \frac{0{,}11 \cdot 2^{11}}{0{,}1 \cdot 2^{10}} = \frac{0{,}11}{0{,}1} \cdot 2^{11-10} = 1{,}1 \cdot 2^1 = 0{,}11 \cdot 2^{1+1} = 0{,}11 \cdot 2^{10}.$$

3: ASCII-Zeichencode

Zeichen	8-Bit-Form (oktale Darstellung)	Zeichen	8-Bit-Form (oktale Darstellung)
!	241	A	301
"	242	B	302
#	243	C	303
$	244	D	304
%	245	E	305
&	246	F	306
'	247	G	307
(	250	H	310
)	251	I	311
*	252	J	312
+	253	K	313
,	254	L	314
—	255	M	315
.	256	N	316
/	257	O	317
:	272	P	320
;	273	Q	321
<	274	R	322
=	275	S	323
>	276	T	324
?	277	U	325
spez. Code	300	V	326
[	333	W	327
\	334	X	330
]	335	Y	331
↑	336	Z	332
←	337	0	260
Zeilen-schaltung	212	1	261
Wagen-rücklauf	215	2	262
Space	240	3	263
Rubout	377	4	264
Form-Feed	214	5	265
Blank	000	6	266
		7	267
		8	270
		9	271

4: SIF-1000-Zeichencode

Zeichen	SIF-1000-Code (oktale Darstellung)	Zeichen	SIF-1000-Code (oktale Darstellung)
0	60	a	141
1	61	b	142
2	62	c	143
3	63	d	144
4	64	e	145
5	65	f	146
6	66	g	147
7	67	h	150
8	70	i	151
9	71	j	152
A	101	k	153
B	102	l	154
C	103	m	155
D	104	n	156
E	105	o	157
F	106	p	160
G	107	q	161
H	110	r	162
I	111	s	163
J	112	t	164
K	113	u	165
L	114	v	166
M	115	w	167
N	116	x	170
O	117	y	171
P	120	z	172
Q	121	Leer-zeichen	40
R	122	!	41
S	123	ε	42
T	124	#	43
U	125		44
V	126	%	45
W	127	&	46
X	130	'	47

Zeichen	SIF-1000-Code (oktale Darstellung)	Zeichen	SIF-1000-Code (oktale Darstellung)
Y	131	(	50
Z	132	)	51
		*	52
+	53	,	54
—	55	.	56
/	57	:	72
;	73	<	74
=	75	>	76
?	77	§	100
[	133	ø	134
]	135	↑	136
\	140	—	137
|	174	NL	36 (Neue Zeile)
¬	176	CR	15 (Wagenrückl.)
		LF	12 (Zeilenschalt.)
		TAB	11 (Tabulator)

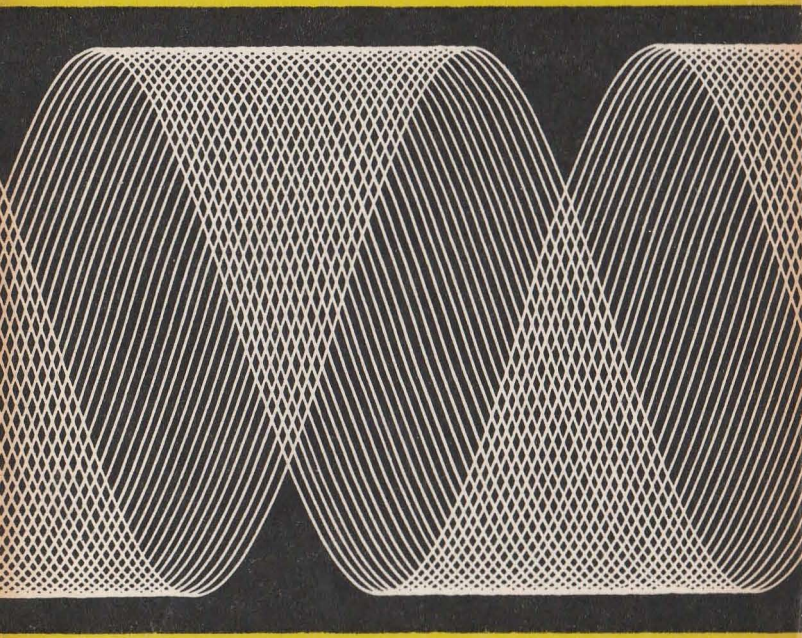
Literaturverzeichnis

[1] *Harro Kühne:* Schaltbeispiele mit TTL-Gattern der Schaltkreisserie D10, Teil 1, Teil 2; Heft 141 und 155 der Reihe *electronica*, Militärverlag der DDR, Berlin 1976, 1977
Manfred Kramer: Impulstechnik mit TTL-Schaltkreisen; Militärverlag der DDR, Berlin 1977

[2] *Eckhardt Groß:* Grundlagen der digitalen Schaltungstechnik, 3. Auflage; Militärverlag der DDR, Berlin 1973

[3] Siemens Datenbuch 1976/77 Mikroprozessor-Bausteine System SAB 8080

[4] *Möschwitzer; Jorke:* Mikroelektronische Schaltkreise; VEB Verlag Technik, Berlin 1979

[5] RFT-Information 1 K bit dynamisches RAM U 253 D

[6] *Kühn, E., Schmied, H.:* Integrierte Schaltkreise; VEB Verlag Technik, Berlin 1972